НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

кафедра автоматика та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ О. І. РОЛІК
(підпис) (ініціали, прізвище)
“___”_____2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 126 «Інформаційні системи та технології»
(код і назва спеціальності)

на тему: Система індексування текстів бази знань

Виконав : студент 6 курсу, групи ІА–з72мп
(шифр групи)

Старченко Артем Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.тех.н. Новацький А.О. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
“Київський політехнічний інститут
імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки
(повна назва)
Кафедра автоматики та управління в технічних системах
(повна назва)
Ступінь вищої освіти –другий (магістерський)
(код, назва)
Спеціальність 126 «Інформаційні системи та технології»
(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О. І. РОЛІК
(підпис) (ініціали, прізвище)

“___”_____2018_р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Старченку Артему Олександровичу

(прізвище, ім’я, по батькові)

1.**Тема дисертації__** Система індексування текстів бази знань

_____

Науковий керівник дисертації__Новацький Анатолій Олександрович, к. т. н., доцент_____

затверджені наказом по університету від “_29_” _жовтня_2018_р.№_____

2.Строк подання студентом дисертації “_4_” грудня 2018_р.

3. Об`єкт дослідження: Система індексування текстів бази знать

4.Зміст пояснювальної записки**:** а) огляд існуючих рішень; б)_____опис поставленної задачі; в_Опис системи індексування текстів;_ г) тестування роботи системи; д) стартап;

5. Перелік графічного (ілюстративного) матеріалу

Алгоритм морфологічного аналізу поточного тексту;_ алгоритм визначення тематики тексту в системі індексування; алгоритм створення індексу І-го рівня; алгоритм створення індексу ІІ-го рівня; діаграма класів; діаграма послідовності попереднього налаштування системи; діаграма специфікацій системи індексування текстів «Основна форма»; структура індексування текстів; use case діаграма.

6. Консультанти розділів проекту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання "_29_" __жовтня__2018_р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Огляд існуючих рішень та алгоритмів	10.11.2018	
2	Опис рішення поставленної задачі	15.11.2018	
3	Опис системи індексування текстів	20.11.2018	
4	Тестування роботи системи	25.11.2018	
5	Стартап	29.11.2018	
6	Оформлення висновків магістерської дисертації	2.12.2018	

Студент _____ __ Старченко А.О.
(підпис) (ініціали, прізвище)

Керівник проекту _____ _ Новацький А.О.

АНОТАЦІЯ

Магістерська атестаційна робота “Система індексування текстів бази знань”: 144 с., 78 рис., 15 табл., 1 додаток, 20 джерел.

Об'єкт дослідження – системи управління знаннями.

Мета роботи – аналіз існуючих алгоритмів індексування текстів в системах управління знаннями і розробка нового механізму індексування текстів.

Метод дослідження – аналіз існуючих механізмів.

В основних методах індексування, класифікаційне та координатне, є деякі недоліки. В першому, тексти в залежності від їх змісту, відносяться до відповідного класу, в якому накопичуються всі тексти, що мають схожий зміст. Кожному такому тексту відповідає індекс даного класу, що і виступає його пошуковим образом. Недоліком є те, що тексти можуть відрізнятися і при цьому мати один загальний індекс.

Алгоритм координатного індексування документів оснований на врахуванні класифікаційних характеристик присутніх в тексті термінів (слів та словосполучень), характеризуючи ту чи іншу предметну область. Для цього необхідно створення словника термінів предметної області, при чому, в цьому словнику повинні бути установлені зв’язки між термінами та проведена їх класифікація. Такий словник називається тезаурусом. Його створення потребує досить великих зусиль, що і є недоліком цього алгоритму.

Для створення індексів текстів розроблено механізм, котрий для кожного документу створює індивідуальний індекс. При проходженні процесу індексування, враховується предметна область тексту, але на відміну від тезаурусу, де необхідно встановлювати зв’язки між термінами, використовуються словники предметних областей.

В основу алгоритму покладено модифіковану математичну модель латентно-семантичного індексування. Окрім синонімії, враховується

морфологічний аналіз кожного слова тексту, що розширює можливості даної моделі.

В процесі дослідження, аналізу та розробки алгоритму використовувався пакет Enterprise Architect. Для реалізації та проведення експериментів написано програмний продукт в пакеті MS Visual Studio 2017 Community Edition на мові C# під платформою .NET. Для збереження обробленої інформації використовується база даних Access.

Прогнозні припущення про розвиток дослідження – при створенні індексу, врахування полісемії та омонімії слів.

ІНДЕКС, ПОШУКОВА СИСТЕМА, СИСТЕМА УПРАВЛІННЯ ЗНАННЯМИ, КООРДИНАТНЕ ІНДЕКСУВАННЯ, КЛАСИФІКАЦІЙНЕ ІНДЕКСУВАННЯ, СЛОВНИКИ ТЕМАТИК, МОРФОЛОГІЧНИЙ АНАЛІЗ, ЕУД, КЛЮЧОВІ СЛОВА, МАТЕМАТИЧНА МОДЕЛЬ

ABSTRACT

Master's degree attestation work of “System of indexing texts in knowledge management system”: 127 pages, 78 pictures, 10 tables, 1 addition, 20 sources.

A research object is knowledge management system.

A purpose of work is an analysis of existent algorithms of indexing texts in knowledge management system and development new mechanism indexing texts.

A research method is an analysis of existent mechanisms.

In basic indexing methods, classification and coordinate, there are some failings. In the first, texts depending on their maintenance, behave to the proper class, which contain all texts that have alike maintenance. All this texts have index of this class, that becomes searching image. Failing is that texts can differ and here have one general index.

The algorithm of the coordinate indexing of documents is based on the account of classification descriptions of present in a text terms (words and combinations of words), characterizing subject of text. For this purpose it is necessary creation dictionary of terms for subject domain, and in this dictionary must be relations between terms and they must be classified. Such dictionary named as thesaurus. His creation needs enough large efforts, that is the lack of this algorithm.

For creation indexes of texts was developed mechanism which creates an individual index for every document. At passing of indexing process, the subject domain is taken place, but unlike thesaurus, where it is necessary to set relations between terms, the dictionaries of subject domains are utilized.

The modified mathematical model of the latently semantic indexing take place in algorithm. Except for synonyms, the morphological analysis of every word take place, which extends possibilities of this model.

In the process of research, analysis and development of algorithm the package of Rational Rose was utilized. For realization and experiments a software product is

written in the package MS Visual Studio 2017 Community Edition in language of C# under platform .NET. For the maintenance of the treated information a database Access is utilized.

Prognosis suppositions about development of research - at creation of index, account of polysemys and homonymy of words.

INDEX, SEARCHING SYSTEM, KNOWLEDGE MANADGEMENT SYSTEM, COORDINATE INDEXING, CLASSIFICATION INDEXING, DICTIONARIES of SUBJECTS, MORPHOLOGICAL ANALYSIS, ECD, KEYWORDS, MATHEMATICAL MODEL

ЗМІСТ

ВСТУП

Актуальність. Сьогодні досвідчені професіонали, що займаються управлінням знаннями стали невід'ємною частиною світу інформаційних технологій. Для будь-якої організації, охочої досягти успіху в сьогоднішній глобальній інформаційній економіці, необхідна інтелектуальна, багатофункціональна і проста у використанні система для управління запасами знань, доступу до знань та їх придбання.

Вміння найефективніше розпорядитися інформаційними ресурсами – одна з основних вимог успішної діяльності будь-якої організації, що працює в умовах постійної конкуренції. Зрештою потрібно говорити про безпеку бізнесу, галузі і держави в цілому.

При цьому кількість інформації, з якою доводитися працювати аналітикам, експертам і керівникам постійно росте.

Досвід більш ніж 1300 професіоналів компанії Reuters показав, що:

– 25% вимагають неймовірні об'єми інформації;

– 38% витрачають значний час на пошуки потрібної інформації;

– 41% вважають, що умови роботи занадто ускладнені;

– 47% вважають, що пошук інформації відриває їх від основної роботи;

– 49% відчувають, що не можуть «переварити» отриману інформацію;

– 94% не вірять в поліпшення ситуації.

Причини впровадження систем управління знаннями:

– зростання доходів і прибутку компанії - 67%;

– бажання утримати ключових працівників і експертів - 54%;

– поліпшення обслуговування клієнтів - 52%;

– захист ринку від вторгнення нових гравців - 44%;

– прискорення часу виводу на ринок нового продукту - 39%;

– проникнення в нові сегменти ринку - 39%;

– зниження витрат - 38%;

– розвиток нових продуктів і послуг – 35% [1].

Підводячи короткий і невтішний підсумок таких досліджень, в звіті ЮНЕСКО за 2017 рік 74% європейського суспільства був поставлений «діагноз»: «Синдром інформаційної втоми – інфофобія».

Діяльність будь-якої організації – пошук необхідних процедурних рішень в системі розподіленої інформації і знань.

Знання організації (або корпоративні знання) – це та багатообразна інформація, яку необхідно мати для підтримки на високому рівні основних бізнес-процесів організації, а також для швидкого і адекватного реагування на різні дії. Це пов'язано з тим, що:

– знання породжуються лише з задач адекватних інформації;

– знання – матеріалізована і необхідна інформація;

– документ – це вже матеріалізована форма необхідної інформації.

Як тільки вийде впровадити той або інший процес розповсюдження і сумісного використання знань, відразу встане питання фільтрації неминучого перенасичення інформації в значущі знання, які можна реально застосувати, тобто постає питання пошуку знань. Пошук знань є вищою формою пошуку інформації, оскільки повинен володіти інтелектуальним доступом до інформації і шукати будь-які типи даних, найбільш адекватних запиту, в будь-якому місці. Саме тому тема магістерської роботи є дуже актуальною.

Об’єкт дослідження – системи управління знаннями.

Предмет дослідження – алгоритми індексації текстів в базах знань.

Мета роботи – провести аналіз існуючих рішень та розробити модифікований алгоритм індексування текстових документів з врахуванням семантики текстів. Для досягнення мети потрібно розв’язати наступні *задачі*:

обрати алгоритми для дослідження, провести компаративний аналіз за існуючими критеріями, розробити модифікований алгоритм індексації текстів.

Для всіх видів пошуку найважливішим є механізм оцінки значущості знайдених документів, щоб в першу десятку із списку знайдених завдовжки в тисячу, потрапили дійсно найбільш корисні. Окрім тривіальних методів на основі частоти використання слів запиту, система повинна враховувати їх розташування в документі, зокрема фізична відстань між словами.

Тому, існує ряд вимог, яких необхідно дотримуватися при створенні системи управління знаннями. В основному ці вимоги стосуються процесу обробки і збереження інформації в базі, а також етапу пошуку.

Перерахуємо ці вимоги:

- смисловий пошук;
- функціональна масштабність і відкрита архітектура;
- багатомовна підтримка;
- підтримка різноманітних форматів документів;
- локалізація областей дослідження;
- висока швидкість пошуку;
- швидка і ефективна обробка (індексування) текстових ресурсів;
- простота інтерфейсу користувача.

Для реалізації цих вимог необхідне використання ефективних механізмів обробки інформації, що зберігається. Існують відомі алгоритми індексування текстів, але вони мають як плюси, так і мінуси [2].

Що стосується області застосування, можна сказати коротко: системи управління знаннями необхідні скрізь, де існують завдання по роботі з великими об'ємами різнорідної інформації, де необхідно оперативно вирішувати завдання управління, дослідження і безпеки. В першу чергу, це відноситься до державного управління і безпеки, великого бізнесу, наукової і дослідницької роботи, практичної роботи в багатьох областях знань [3].

Практична цінність. Який ефект на підвищення продуктивності праці надала технологія управління знаннями (результати опиту директорів по розвитку бізнесу) [4]:

– покращує робочі процеси такою мірою, що кількість виникаючих завдань скорочується – 84%;

– сприяє пошуку і розвитку нових технологій – 82%;

– знімає бар'єри між різними робочими процесами – 75%.

Виходячи зі всього вищесказаного можна зробити висновки, що розвиток даного напряму на українському ринку принесе незаперечну користь не тільки процвітанню бізнесу, але і економці країни в цілому.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА АЛГОРИТМІВ

1.1 Сутність проблеми

Управління знаннями – це систематичні процеси, завдяки яким розпізнаються, створюються, зберігаються, розподіляються і застосовуються необхідні для успіху організації знання.

Після впровадження процесу розповсюдження і сумісного використання знань, відразу постає питання фільтрації інформації в значущі знання, які можна реально застосувати, тобто питання пошуку знань. Пошук знань є вищою формою пошуку інформації, оскільки повинен виконуватися інтелектуальний доступ до інформації та пошук будь-яких типів даних, найбільш адекватних запиту.

Під час пошуку необхідної інформації виникає безліч труднощів, при роботі з даними у великих інформаційних просторах.

По-перше, ресурси Інтернет часто не вирішують всього спектру завдань користувачів – знайти щось потрібне за допомогою звичайних пошукових систем буває нелегко і вимагає великих витрат часу. Окрім цього, для будь-якої організації надзвичайно важлива можливість використовувати власний досвід роботи в конкретних областях діяльності. Як результат - практично кожна організація створює власний електронний архів даних (найчастіше в змішаному варіанті використання внутрішніх і зовнішніх інформаційних ресурсів).

По-друге, зазвичай інформація знаходиться в неструктурованому вигляді.

По-третє, виникає маса проблем при роботі з документами на різних мовах.

Тому, існує ряд вимог, яких необхідно дотримуватися при створенні системи управління знаннями. В основному ці вимоги стосуються процесу обробки і збереження інформації в базі, а також етапу пошуку. Одним з

основних і є модуль що реалізовує ці вимоги. Тому, дана робота присвячується розробці механізму обробки документів (індексування) в системах управління знаннями.

Перерахуємо необхідні вимоги:

1. Смисловий пошук – в даному сенсі має місце морфологічний аналіз, який використовується, як при обробці строки запиту, так і при створенні індексу документа.

2. Логічний (булевий) пошук забезпечує високу точність витягання інформації по специфічних запитах, але вимагає від користувача знання предмету дослідження.

3. Функціональна розширюваність і відкрита архітектура для систем управління знаннями є загальноприйнятими. Тільки відкрита система може надавати розробникам можливості адаптації системи до умов місцевого ринку, а також тонкої настройки системи під вимоги замовника. Вимоги відкритості архітектури системи повинні розповсюджуватися на всі модулі системи аж до ядра.

4. Багатомовна підтримка. Сьогоднішні спеціалісти володіють багатьма мовами, корпоративні документи часто оформляються на різних мовах. Тому підтримка багатьох багатомовності є невід'ємною частиною вимог до системи.

5. Підтримка різноманітних форматів документів – необхідна вимога до системи управління знаннями.

6. Локалізація областей дослідження дозволить мінімізувати загальноприйняті слова під час роботи системи. Використання спеціалізованих словників по різних предметних областях надає можливість користувачам проводити пошук в окремих сферах не плутаючи специфічні терміни.

7. Висока швидкість пошуку – важливий критерій оцінювання пошукових систем.

8. Швидка і ефективна обробка (індексування) текстових ресурсів.

9. Простота інтерфейсу користувача.

Для реалізації більшості з цих вимог необхідно використовувати ефективні механізми обробки інформації.

Існують відомі алгоритми індексування текстів, але вони мають як переваги, так і недоліки.

При класифікації, тексти залежно від їх змісту відносяться до відповідного класу (одного чи декількох), в якому збираються всі тексти, що мають в основному однаковий смисловий зміст. Кожному такому тексту привласнюється індекс цього класу, що слугує далі його пошуковим образом.

Модифікований алгоритм індексування документів заснований на врахуванні класифікаційних ознак присутніх в тексті термінів (слів і словосполучень), що характеризують ту або іншу предметну область. Для цього необхідне створення словника термінів предметної області, причому в цьому словнику повинні бути встановлені зв'язки між термінами і проведена їх класифікація. Такий словник називається тезаурусом.

В основу розроблюваного алгоритму покладено модифіковану латентно-семантичну модель. Модифікація заключається в тому, що окрім синонімії та полісемії, враховується різновид словоформ кожного слова.

1.2 Аналіз існуючих рішень

При аналізі існуючих рішень було розглянуто ряд систем, в яких одним з основних завдань є обробка текстової інформації, також досліджені основні алгоритми індексування та математичні моделі пошуку, серед них :

1) системи управління знаннями: IBM Infosphere Information Server [5]; Open Text Livelink ECM [6];

2) системи управління документообігом: Логика «СЭД» [7], Docsvision [8], Directum [9], та інші.

3) пошукові системи: NEUROK Semantic Suite [10]; diskmeta-lite [11]; Mtsearch [12];

4) види індексування: класифікаційне індексування; координатне індексування; індексування документів пошуковими роботами; та інші;

5) математичні моделі: булева; векторна; імовірнісна та латентно-семантична.

1.2.1 Системи управління знаннями

Управління знаннями - це систематичні процеси, завдяки яким розпізнаються, створюються, зберігаються, розподіляються і застосовуються необхідні для успіху організації знання.

1.2.1.1 IBM Infosphere Information Server

Пакет Infosphere Information Server фірми IBM повністю автоматизує управління знаннями і документами і є найбільш всеосяжним рішенням в цьому огляді. Програма відстежує всю інформацію, необхідну для повсякденної роботи користувачів.

Дані з численних джерел, таких як: електронна пошта, файл-сервери, бази даних і платформи колективної роботи, зберігаються автоматично без втручання користувача.

На основі цих даних Infosphere Information Server будує коротке представлення знань і обов'язків користувача. Більш того, пакет створює профілі користувачів на базі документів, які вони проглядають і пишуть, а

також відстежує персональну інформацію, таку як переваги, навички, освіта і посадові обов'язки.

1.2.1.2 Open Text Livelink ECM

Пакет Livelink компанії Open Text включає декілька інструментів управління знаннями і платформу колективної роботи. Управління контентом, документами і записами - найбільш важливі функції управління знаннями. Livelink також об'єднує функції документообігу і автоматизації бізнес-процесів, інтегруючи програми і корпоративні додатки інших фірм.

Адміністратор може імпортувати систему класифікації з XML-файла або створити її «з нуля». Коли класифікації або категорії вводяться в програму, створюється профіль, який охоплює всі правила, використані для виявлення ключових фраз. Крім того, правила можуть бути явно задані користувачами. Livelink забезпечує доступ до профілів, так що правила в запитах на пошук можуть формувати критерії для витягання фраз.

Для пошуку можна використовувати логічні вирази, ключові фрази, запити на природній мові з документів. Крім того, Livelink здатний створити на основі фраз, що виявлені в документі, його короткий зміст, який можна включати в пошук як окремий елемент. Існує можливість виконувати пошук відразу по декількох репозиторіях, зокрема тим, що належать деяким іншим пакетам управління документами і контентом. Пакет дозволяє використовувати широкий спектр атрибутів і комбінацій пошуку, тому немає ніяких обмежень на завдання правил. Наприклад, для пошуку можна ввести ключову фразу разом з ім'ям автора документа [6].

1.2.2 Електронні системи управління документами

Електронні системи управління документами (ЕУД) – це системи, які забезпечують процес створення, управління доступом і розповсюдження великих об'ємів документів в комп'ютерних мережах, а так само контроль над їх потоками в організації. В основному, документи зберігаються в базі даних.

В сучасних розподілених підприємствах системи розповсюдження документів, вимоги по їх доступності і необхідність спільної роботи з ними ростуть експоненціально. Інформаційні матеріали, що мають високу цінність створюються щодня, розміщуються в глобальних мережах, розповсюджуються в різних професійних колективах.

1.2.2.1 Можливості систем ЕУД

Щоб отримати уявлення про можливості систем, що дозволяють управляти електронним документообігом, представлена таблиця з необхідним набором функціоналу (таблиця 1.1). Як показано в таблиці, всі представлені системи володіють майже всіма необхідними функціональними можливостями.

В таблиці 1.1 і в наступних використовуються скорочення: Бос-Референт (I), Грандок (II), Справа (III), ЕВФРАТ-Документообіг (IV), Companymedia (V), Directum (VI), DIS-системы (VII), Docsvision (VIII), Landocs (IX), Optima-workflow (X). Також, в таблицях: “+“ – означає що опція присутня, “-“ – опція відсутня, “+/-“ - опционально або зовнішнім ПЗ, “-/+“ - неповна функціональна можливість.

Таблиця 1.1 – Основні можливості систем електронного документообігу

Система / Опція	I	II	III	IV	V	VI	VII	VIII	IX	X
Реєстрація документів	+	+	+	+	+	+	+	+	+	+
Ведення реєстраційної картки	+	+	+	+	+	+	+	+	+	+

Продовження таблиці 1.1

Ведення номенклатури справ	+	+	+	+	+	+	+	+	+	+
Сканування	+/-	+	+/-	+	+/-	-/+	+	+	+	+/-
Розпізнавання документів	+/-	+	+/-	+	+/-	-/+	+	+	+	+/-
Об’єднані документи	+	+	+	+	+	+	+	+	+	+
Прикріплені документи	+	+	+	+	+	+	+	+	+	+
Робота зі словниками та довідниками	+	+	+	+	+	+	+	+	+	+
Строки поручень	+	-	+	+	+	+	+	+	+	+
Пошук:										
- по реквізитам	+	+	+	+	+	+	+	+	+	+
- по виду РК документу	-	-	+	+	-	-	+	+	-	+
-повнотекстовий	+	+	+	+	+	+	-	+	+	+
- з урахуванням морфології	-	-	-	+	-	+	-	-	+	+/-
Списання документу в архів	+	+	+	+	+	+	-	+	+	+
Ведення архівів електронних документів	-/+	-	+/-	+/-	-/+	-/+	-	-/+	+/-	-/+
Маршрутизація	+	-	+	+	+	+	-	+	+	+
Генерація звітів	+/-	+/-	+	+	+/-	+	+	+	+	+/-
Розмежування прав доступу	+	+	+	+	+	+	+	+	+	+
Ролі	+	-	-/+	+	+	+	+	+	+	+

1.2.2.2 Додаткові настройки систем ЕУД

У таких систем є безліч складнощів, як в процесі впровадження, так і в процесі експлуатації. Наприклад, якщо придбана система, не містить

інструментарію необхідного для настройки під потреби організації власними силами, то процес впровадження вимагатиме участі розробника, що істотно вплине на підвищення вартості впровадження. Можливість адаптації системи під потреби конкретної організації без участі розробника і без додаткових витрат передбачена до ЕВФРАТ-Документообіг і Docsvision. Власникам інших систем пропонується набір стандартних настройок, які можуть бути допрацьовані розробниками або їх партнерами, що приведе до збільшення витрат на впровадження (таблиця 1.2).

Таблиця 1.2 – Доступний інструментарій для настройки системи

Система / Інструментарій	I	II	III	IV	V	VI	VII	VIII	IX	X
Дизайнер форм карточок документів	+	-	-/+	+	+/-	-/+	-	+	-/+	+
Дизайнер маршрутів/процесів	+	-	-/+	+	+/-	-/+	-	+	+	+
Редактор звітів	+	+	-	+	+	-/+	-	+	+	+
Створення та редагування словників і довідників	+	+	-	+	+	-/+	-	+	+	+

У кожній такій системі існує додаткові опції, які не представлені в таблиці 1.2. Ці опції значно спрощують роботу з системою всім співробітникам організації (наприклад, настройка повідомлень і нагадувань). В деяких випадках певний функціонал просто необхідний (наприклад, WEB-доступ). Потреби замовника в автоматизації процесів обробки документів з часом можуть мінятися (наприклад, у разі зміни організаційної структури підприємства), що може привести до необхідності в розвитку системи.

Як видно з таблиці 1.3, найбільш повноцінний додатковий функціонал за рахунок гнучких настройок призначеного для користувача інтерфейсу надають системи Бос-Референт і Optima-workflow.

Таблиця 1.3 – Додаткові можливості роботи з системою

Система Можливість	I	II	III	IV	V	VI	VII	VIII	IX	X
Настройка інтерфейсу користувача	+	-/+	-/+	-/+	+/-	-/+	-/+	-/+	-/+	+
Настройка повідомлень та нагадувань	+	-/+	-/+	+	+	+	-/+	+	+	+
Інтеграція з електронною поштою	+	-	-/+	+	+	+	-/+	+	+/-	+
WEB-доступ	+	-	+/-	+	+/-	+/-	+	+	+/-	+

Важливими характеристиками систем управління документами є їх безпека, надійність і продуктивність (таблиця 1.4). З таблиці видно, що системи Directum, Docsvision, Optima-workflow і ЕВФРАТ-Документообіг дозволяють забезпечити необхідний рівень надійності і безпеки електронного документообігу.

Таблиця 1.4 – Надійність і безпека

Система Можливість	I	II	III	IV	V	VI	VII	VIII	IX	X
Авторизація користувачів з паролем	+	+	+	+	+	+	+	+	+	+
Шифрування документів	+/-	-	+/-	+	+/-	+	+	+	+/-	+
Резервне копіювання по розкладу	+	-	+	+	+	+	-	+	+	+

Найкращим чином з типовими завданнями електронного документообігу справляються системи: Дело, ЕВФРАТ-Документообіг, Docsvision і Landocs.

1.2.3 Огляд пошукових систем

Пошукова система - це система, що дає можливість пошуку інформації. Існує безліч класифікацій таких систем: одні призначені для пошуку в Інтернеті, інші - на жорсткому диску комп'ютера, треті - в базі даних. На один і той же запит в різних пошукових системах, відповідь відрізняється. Це залежить від алгоритмів закладених в структуру роботи системи.

Порядок роботи алгоритмів необхідний пошуковим системам для визначення релевантності відповідно до запиту користувача. Алгоритм пошукової системи розглядається як безліч математичних формул. Алгоритм використовує ключові запити і надає релевантні результати у вигляді вирішення задач. Ключові запити визначаються пошуковими роботами, де перевіряється контент сторінки і релевантність запитів на основі формул алгоритмів, які у кожної пошукової системи різні.

Є сервіси, які збирають інформацію про запити, що часто зустрічаються, про сторінки, що найчастіше переглядаються, і час витрачений на кожну сторінку. Отримана інформація застосовується для видачі результатів про запити, які найпопулярніші у користувачів. Безліч запитів до яких застосована ця технологія несе за собою спам.

Ці бази даних створювалися на основі згрупованої користувачем інформації. Даний метод розглядається як архаїчний, хоча існує не мало директорій, складових бази пошукових систем, такі як Open Directory і DMOZ, які групуються вручну. Матеріали в деяких пошукових системах формується вручну, як тільки пошукові роботи зберуть необхідну інформацію. Алгоритми

аналізують розташування ключових слів, сторінки з високою частотою сприймаються як більш релевантні, це називається щільність ключових слів.

1.2.3.1 Модель роботи пошукових систем

Модель пошукової системи представлена на рисунку 1.1 [13].

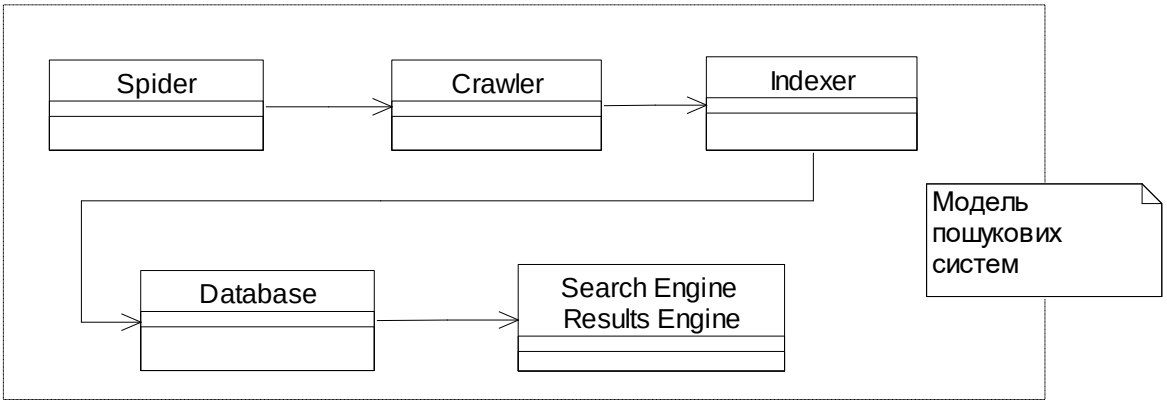


Рисунок 1.1 – Модель роботи пошукової системи

Пошукові системи складаються з п'яти окремих програмних компонент:

1. spider (павук): програма, яка викачує веб-сторінки. Він працює точно як браузер, коли відбувається з'єднання з веб-сайтом і завантаження сторінок. Павук не має ніяких візуальних компонент. Ту ж дію (скачування) ми спостерігаємо, коли проглядаємо деяку сторінку і коли вибираємо "перегляд HTML-кода" в своєму браузері.

2. crawler : "мандрівний" павук, який автоматично йде по всіх посиланнях, знайдених на сторінці. Як і павук, викачує сторінки, аналізує сторінку і знаходить всі посилання. Це його завдання - визначати, куди далі повинен йти павук, ґрунтуючись на посиланнях або виходячи із заздалегідь заданого списку адрес.

3. indexer (індексатор): "сліпа" програма, яка аналізує веб-сторінки, що скачали павуки. Індексатор розбирає сторінку на різні її частини і аналізує їх.

Елементи типу: заголовків сторінок, посилань, тексту, структурних елементів, і інших стильових частин сторінки виділяються і аналізуються.

4. the database (база даних): сховище скачаних та оброблених сторінок. База даних - це сховище всіх даних, які пошукова система викачує і аналізує. Це часто вимагає величезних ресурсів.

5. search engine results engine (система видачі результатів): витягує результати пошуку з бази даних. Саме система видачі результатів вирішує, які сторінки задовольняють запиту користувача. Це та частина пошукової системи, з якою ми маємо справу здійснюючи пошук.

На рисунку 1.2 перераховано декілька алгоритмів пошуку текстів.

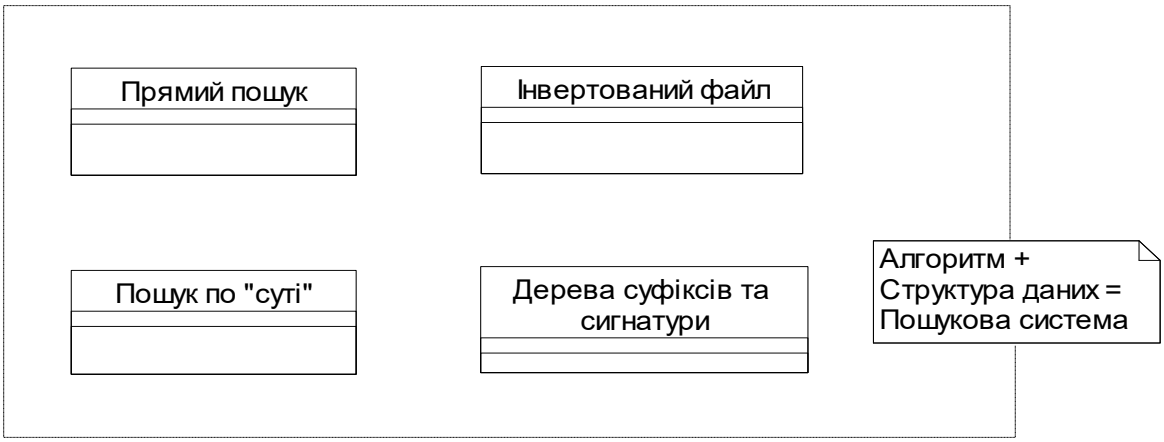


Рисунок 1.2 – Перелік основних алгоритмів пошукових систем

1.2.3.1.1 Інвертований файл

Припустимо, є впорядкований за абеткою список слів. Для кожного слова перераховані всі "позиції", в яких це слово зустрілося. Пошуковий алгоритм полягає у відшуканні потрібного слова і завантаженні в пам'ять вже розгорненого списку позицій.

Щоб заощадити на дисковому просторі і прискорити пошук, зазвичай удаються до двох прийомів. По-перше, можна заощадити на подробицях самої

позиції. Адже чим докладніше задана така позиція, тим більше місця буде потрібно для зберігання інвертованого файлу.

В найчіткішому варіанті в інвертованому файлі можна зберігати і номер слова, і зсув в байтах від початку тексту, і колір і розмір шрифту, та багато чого ще. Частіше ж просто указують тільки номер документа, і число вживань цього слова в ньому. Саме така спрощена структура вважається основною в класичній теорії інформаційного пошуку.

Другий (ніяк не пов'язаний з першим) спосіб стиснення: упорядкувати позиції для кожного слова за збільшенням адрес і для кожної позиції зберігати не повну її адресу, а різницю від попередньої.

Додатково до попереднього, на спосіб зберігання адрес накладають простий метод архівації: навіщо відводити невеликому цілому числу фіксовану "величезну" кількість байт, адже можна відвести йому стільки байт, скільки воно являє собою.

В результаті всіх описаних способів розмір інвертованого файлу, як правило, складає від 7 до 30 відсотків від розміру початкового тексту, залежно від подробиць адресації.

1.2.3.1.2 Прямий пошук

Прямий пошук – пошук безпосередньо по тексту документів, без попередньої обробки.

Останні 30 років прямий пошук інтенсивно розвивається. Було висунуто чимало ідей, що скорочують час пошуку в рази. Ці алгоритми детально описані в різноманітній літературі, є їх зведення і порівняння. При цьому треба врахувати, що нові алгоритми і їх оновлені варіанти з'являються постійно.

Хоча прямий перегляд всіх текстів - досить повільне заняття, не слід думати, що алгоритми прямого пошуку не застосовуються в Інтернеті.

Норвезька пошукова система Fast використовувала мікросхему, що реалізовує логіку прямого пошуку спрощених регулярних виразів, і розмістила 256 таких мікросхем на одній платі. Це дозволяло Fast-у обслуговувати досить велику кількість запитів в одиницю часу.

Крім того, є маса програм, що комбінують індексний пошук для знаходження блоку тексту з подальшим прямим пошуком всередині блоку.

Взагалі, у прямих алгоритмів принципове безпрограшні відмінні риси. Наприклад, необмежені можливості по наближеному і нечіткому пошуку. Адже будь-яке індексування завжди зв'язане із спрощенням і нормалізацією термінів, а, отже, з втратою інформації. Прямий же пошук працює безпосередньо по оригінальних документах без всяких спотворень [14].

1.2.3.1.3 Дерева суфіксів та сигнатури

Неодноразово пропонувалися інші, відмінні від інвертованого і прямого пошуку алгоритми і структури даних. Це, перш за все, дерева суфіксів, та сигнатури.

Перший з них функціонував в Інтернеті, будучи запатентованим алгоритмом пошукової системи Opentext.

Другий – метод сигнатур - є перетворення документа до блочних таблиць значень його слів - "сигнатури" і послідовного перегляду "сигнатур" під час пошуку [14].

Широкого розповсюдження ні той ні інший метод не отримали.

1.2.3.1.4 Семантичний пошук

Здатність знаходити документи, що не містять слів із запиту, часто вважають ознакою штучного інтелекту або пошуку по суті і відносять апріорі до переваг моделі.

У теорії інформаційного пошуку дану модель прийнято називати латентно-семантичним індексуванням (іншими словами, виявленням прихованих значень). Ця модель алгебри заснована на сингулярному розкладанні прямокутної матриці, що асоціює слова з документами.

Сингулярним розкладанням дійсної матриці A розмірів m*n називається всяке її розкладання виду A = USV, де U - ортогональна матриця розмірів m*m, V - ортогональна матриця розмірів n*n, S - діагональна матриця розмірів m*n, елементи якої sij= 0, якщо і не рівне j, і sij = si >= 0. Величини si називаються сингулярними числами матриці і рівні арифметичним значенням квадратного коріння з відповідних власних значень матриці AAT. У англомовній літературі сингулярне розкладання прийнято називати SVD-розкладанням.

Операції пошуку або знаходження схожих документів різко спрощуються, оскільки кожному слову і кожному документу ставиться в відповідність відносно короткий вектор з k сенсів (рядки і стовпці відповідних матриць). Але чи внаслідок малої суті "сенсів", або по якій іншій причині, використання даної моделі для пошуку так і не набуло поширення. Хоча в допоміжних цілях (автоматична фільтрація, класифікація, розділення колекцій, попереднє пониження розмірності для інших моделей) цей метод знаходить застосування.

Проте, яка б не була модель, пошукова система потребує оцінки якості пошуку і настройки параметрів. Оцінка якості - ідея, фундаментальна для теорії пошуку. Бо саме завдяки оцінці якості можна говорити про застосовність або не застосовність тієї або іншої моделі і навіть обговорювати їх теоретичні аспекти [14].

1.2.4 Огляд існуючих програмних продуктів пошукових систем

Розглянемо ряд програмних продуктів, що виконують інформаційний пошук:

- NEUROK Semantic Suite;
- diskmeta-lite;
- Mтsearch.

1.2.4.1 NEUROK Semantic Suite

Сімейство продуктів NEUROK Semantic Suite застосовує технології машинного навчання і розпізнавання образів для роботи з текстовою інформацією. Семантичний сервер NEUROK Semantic Suite забезпечує автоматичне виявлення базових семантичних категорій і індексує з їх допомогою документи відповідно до їх змісту.

Будь-який уривок тексту, аж до окремого терміну, можна розглядати як образ, сенс якого визначається набором смислових (семантичних) категорій. Виявлення системи семантичних категорій відбувається на етапі попереднього навчання системи - в процесі категоризації. Для цього потрібна достатньо чітка вибірка документів з тієї предметної області, в якій передбачається використовувати Semantic Suite. Використовуючи виявлені в процесі навчання знання про семантику термінів, останній організовує свій внутрішній асоціативний індекс оптимальним чином - для швидкого доступу до індексованої інформації по її змісту.

Кожен документ, розкладений за системою семантичних категорій, можна розглядати як крапку в семантичному просторі, а колекцію документів - як набір крапок. NEUROK Semantic Suite здійснює ієрархічну кластеризацію колекцій документів, автоматично виявляючи їх тематичну структуру для зручності навігації і пошуку інформації.

Тематична структура колекцій, загалом, відображається в інтерфейсі компоненти NEUROK Semantic Explorer у вигляді тематичного дерева [10] .

1.2.4.2 diskmeta-lite

Програма diskmeta – призначена для пошуку файлів на комп'ютері і дозволяє миттєво знаходити текстові документи у всіх популярних форматах.

В таблиці 1.2 описані основні можливості системи пошуку diskMETA-Lite [11].

Таблиця 1.2 – Основні можливості системы diskMETA-Lite.

Характеристика	Значення
Средняя швидкість пошуку	100 ГБ/год
Средний об’єм індексу	25% від початкових документів
Середня швидкість пошуку	1 сек
Формати текстів	Залежить від версії програми
Лінгвістичні функції	Врахування словоформ для російської мови
Алгоритм сортування результатів	- врахуванні кількості слів з запиту в документі - врахування місця розміщення слова запиту в документі - врахування близькості слів запиту в документів - врахування частоти повторення слова в індексі - врахування дати документу
Видача результатів	- по відповідності запиту - по даті
Відображення результатів	перегляд тексту документу з можливістю переходу по словам запиту
Мова запитів	- логічні оператори - оператори дати - оператори близькості

1.2.4.3 Mtsearch

Mtsearch - пошукова система, що дозволяє здійснювати повнотекстовий пошук документів на російській, українській, англійській та інших мовах в мережах Internet та Intranet, що працюють на платформах Windows.

Mtsearch забезпечує нові можливості по управлінню, пошуку і аналізу текстових даних, використовуючи продукт Microsoft Internet Explorer або продукти Microsoft Office System.

Mtsearch розширює функціональні можливості вбудованих пошукових систем Internet Explorer, Microsoft Office System і Sharepoint Portal Server, дозволяючи збільшити не тільки кількість індексованих ресурсів, але і функціональність завдяки обліку морфологічних особливостей російської та української мов [12].

У таблиці 1.3 представлений порівняльний аналіз існуючих програмних продуктів що реалізовують пошук текстової інформації.

Таблиця 1.3 – Порівняльний аналіз існуючих програмних продуктів виконуючих пошук текстової інформації

Програмний продукт	Швидкість індексу-вання (Gb/год)	Средній об'єм інденксу	Середня швидкість пошуку (сек)	Формати	Клієнт-серверна архітек-тура	Особливості пошуку
NeurOK Semantic Engine	18-20	15-25%	4-6	HTML, XML, MS Office, PDF та інші.	+	Забезпечує автоматичне виявлення базових семантичних категорій та індексує з їх допомогою документи в відповідності з їх змістом
diskMETA-Lite	100	25%	1	Залежить від версії програми.	+	При пошуку враховується не лише кількість повторень слова в документі, а й форма та місце знаходження, елементи форматування і т д.
MTSearch	50	20-25%	1-5	Підтримує майже всі розповсюджені формати документів.	+	Має можливість обмежувати область пошуку з допомогою стандартних атрибутів документів. Таких як: автор, дата, формат файлу і т. д.

1.2.5 Індексування

Всі розглянуті вище системи обробляють великі об'єми текстової інформації. У кожній з них використовується процес індексування.

Індексування, процес виразу головного предмету або теми тексту якого-небудь документа в термінах інформаційно-пошукової мови. Застосовується для полегшення пошуку необхідного тексту серед множини інших. Проводиться, індексування, як цілого документа, так і його частини. Для цього часто використовуються заголовки текстів. При індексуванні не враховуються супутні предмети або теми. Це слугує причиною того, що при пошуку не знайденими залишаються тексти, для яких предмет або тема інформаційного запиту є не головною, а супутньою.

1.2.5.1 Класифікаційне та координатне індексування

Розрізняють декілька механізмів індексування. Одні з них - класифікаційне і координатне.

При класифікаційному, або класифікації, тексти залежно від їх змісту відносяться до відповідного класу (одного чи декількох), в якому збираються всі тексти, що мають в основному однаковий смисловий зміст. Кожному такому тексту привласнюється індекс цього класу, що слугує далі його пошуковим образом.

Алгоритм координатного індексування документів заснований на врахуванні класифікаційних ознак присутніх в тексті термінів (слів і словосполучень), що характеризують ту або іншу предметну область. Для цього необхідне створення словника термінів предметної області, причому в цьому словнику повинні бути встановлені зв'язки між термінами і проведена їх класифікація. Такий словник називається тезаурусом. Списком ключових слів і словосполучень для тезауруса пропонується використовувати предметний покажчик спеціалізованої енциклопедії (або декількох

енциклопедій). Вибір конкретної енциклопедії здійснює фахівець з даної області, і цей вибір залежить від цілей, що переслідуються при створенні тезауруса. Дескрипторами (тобто термінами, класами близькими по сенсу понять) вважаються назви статей енциклопедій, а пов'язаними з ними по сенсу вважаються слова з предметного покажчика, що зустрічаються у відповідних статтях. Основною перевагою такого методу є те, що для установлення типів зв'язків між термінами не потрібно бути експертом в даній області - цілком вистачить загальних знань, що дозволяють зрозуміти текст енциклопедії, - конкретніші відомості, необхідні в процесі класифікації понять, завжди можна почерпнути з відповідної статті.

Далі проводиться класифікація дескрипторів відповідно до розділів даної предметної області. Вибір конкретного класифікатора, як і вибір енциклопедії, здійснюється фахівцем-експертом, причому, у разі використання декількох енциклопедій з різних наочних областей, можливе використання декількох спеціалізованих класифікаторів. Після цього ключовим словам, пов'язаним з дескриптором, приписується той же класифікаційний номер, що і дескриптору. Втім, це не виключає такої ситуації, коли дескриптор віднесений до класу не найнижчого рівня, а при подальшому аналізі експертом термінів, пов'язаних з дескриптором, їх буде віднесено до класу нижчого рівня. В цьому випадку вказані терміни самі стануть дескрипторами.

В результаті всі терміни, що входять в предметний покажчик, виявляються розподіленими відповідно до розділів даної області.

Проте, процес побудови тезаурусу відповідно до даної методики має потребує великого об'єму рутинної роботи і, крім того, вимагає участі людини, що має навики програмування [15].

1.2.5.2 Індексування документів пошуковими роботами

Як відомо, одними з найбільш ефективних і результативних засобів розкручування веб-сайтів є реєстрація в каталогах Інтернет-ресурсів і індексування пошуковими системами. Але якщо перший метод в більшості випадків залежить виключно від того, як влаштована процедура занесення інформації про ресурс в базу даних каталогу, то підхід користувача до індексування веб-документів пошуковими системами з повною впевненістю можна назвати індивідуальним і цілком передбаченим. Реєструючись в каталозі, користувач вносить до встановлених форм дані про сайт так. Проте, далеко не завжди інформація про ресурс буде відображена згодом в каталозі в первозданному вигляді: адміністратори багатьох подібних серверів по-своєму редагують опис ресурсів, керуючись при цьому власними навичками і правилами. На рисунку 1.3 представлені основні кроки обробки документа для його індексування [16].

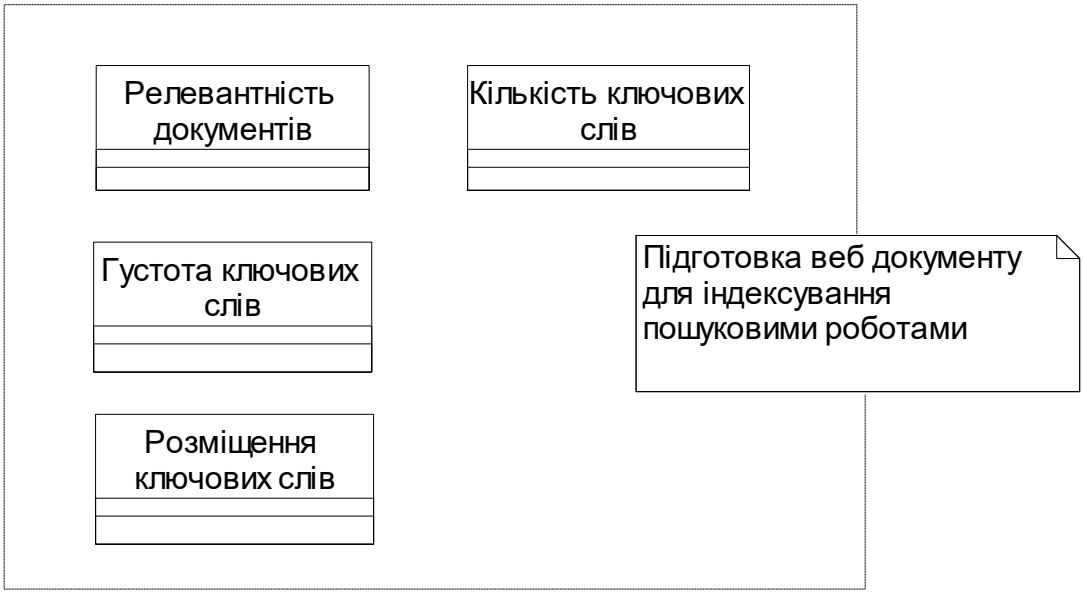


Рисунок 1.3 – Основні кроки обробки веб-документу для індексування пошуковими роботами

1.2.5.2.1 Релевантність документів

Коли користувач вводить в строку запиту пошукової системи якесь слово або словосполучення, робот звертається до всіх проіндексованих сторінок. Кількість отриманих результатів може досягати десятків і навіть сотень тисяч. Робот виводить результати по критерію найбільшої відповідності змісту документів запиту користувача, який називається релевантністю. Іншими словами, самі відповідні сторінки будуть розміщені на початку списку, що видається пошуковою системою. На релевантність впливають чинники ключових слів, мова про яких і піде.

1.2.5.2.2 Кількість ключових слів

Під кількістю ключових слів мається на увазі їх частота присутності в документі. Тобто сторінка, на якій робот виявить 15 разів слово запиту, буде більш релевантна, ніж та, яка містить це слово всього 3 рази. Логічно допустити, що якщо сторінка називається "Кращі безкоштовні програми для Unix", в її тексті навряд чи можна зустріти слова "еротика", "макіяж" або "трубопровід". Зате "система" або "Unix" можуть зустрітися десятки разів.

1.2.5.2.3 Частота ключових слів

Під частотою ключових слів прийнято розуміти ступінь відношення кількості ключових слів до решти слів в межах документа. Пошукові роботи вважають більш релевантною сторінку з конкретним словосполученням, ніж документ, в якому є крім цього словосполучення інші слова і фрази. Наприклад, документ, що містить тільки два слова "комерційна пропозиція", йтиме в списку попереду сторінки, що містить крім поєднання слів "комерційну пропозицію" ще і інші слова.

У таблиці 1.4 приведені основні алгоритми індексування, розглянуті їх переваги та недоліки.

Таблиця 1.4 – Види індексування, їх переваги і недоліки.

Види індексування	Переваги	Недоліки
Координатне	- точність індексування; - створення підказок для строки запиту.	- мала швидкість пошуку; - складність врахування зв’язків між словами в тезаурусі
Класифікаційне	- велика швидкість пошуку; - малий розмір індексу.	- низький рівень точності індексу; - присвоєння текстам з різним змістом спільного індексу
Пошукові роботи	простота реалізації за рахунок прямого перебору текстів на визначення ключових термінів. ключевых слов.	помилки в створенні індексів веб-документів.

1.2.6 Математичні моделі пошуку

Все різноманіття моделей традиційного інформаційного пошуку прийнято ділити на три види: теоретико-множинні (булева, нечітких множин, розширена булева), алгебраїчні (векторна, узагальнена векторна, латентно-семантична) та імовірнісні.

1.2.6.1 Булева модель

Булева модель основана в наступному: є слово – документ вважається знайденим, немає - не знайденим. Власне, класична булева модель - це місток, що зв'язує теорію інформаційного пошуку з теорією пошуку і маніпулювання даними.

Визначення. В булевій моделі, вага індекс терму може приймати лише два значення: $w_{ij} \in \{0,1\}$. Запит q – це булевий вираз. Нехай, $\vec{q}_{dnf}$ - диз'юнктивна нормальна форма запиту q, а $\vec{q}_{cc}$ - кон'юнкція компоненту $\vec{q}_{dnf}$. Тоді, релевантність документу d_j до запиту q визначається формулою (1.1):

$$sim(d_j, q) = \begin{cases} 1 \;\; if \; \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall_{ki}, g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ \hline 0 \;\; документ \;\; є \;\; не \; релевантним \end{cases} \tag{1.1}$$

Отже, якщо $sim(d_j, q) = 1$, то виходячи з принципів мулевої моделі, документ d_j являється релевантним до запиту q. Інакше, він є не релевантним [14].

1.2.6.2 Векторна модель

Ранжирування в цій моделі засноване на природному статистичному спостереженні, що чим більше локальна частота терміну в документі і більше «рідкість» терміну в колекції, тим вище вага даного документа по відношенню до терміну.

Визначення. Для векторної моделі, $w_{i,j}$ - це вага зв'язана з парою (k_i, d_j), котра завжди позитивна і має не бінарну форму. Також, індекс терми в запиті завжди мають вагу. Нехай, $w_{i,q}$ - це вага зв'язана з парою $[k_i, q]$, где $w_{i,q} \geq 0$. Тоді, вектор запиту $\vec{q}$ визначається як $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$, де t – це загальне число індекс термів в системі. Що до вектору документу d_j, то він має наступний вигляд $\overrightarrow{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$.

Векторна модель пропонує встановити залежність між величиною схожості документа d_j відносно запиту q. Ця залежність може бути мірою, наприклад, косинусом кута між двома векторами. Тоді маємо:

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \times \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w^2_{i,j}} \times \sqrt{\sum_{i=1}^{t} w^2_{i,q}}},$$

(2.2)

де $\left|\vec{d_j}\right|$ и $\left|\vec{q}\right|$ нормалі векторів документу та запиту. Фактор $\left|\vec{q}\right|$ не впливає на клас (мається на увазі класифікація документу), тому що він один і той же для всіх документів. Фактор $\left|\vec{d_j}\right|$ вводить ранжування в простір документів.

Якщо $w_{i,j} \geq 0$ і $w_{i,q} \geq 0$, то $sim(q, d_j)$ лежить в межах від 0 до +1. Таким чином, незалежно від спроби спрогнозувати, чи являється документ релевантним, векторна модель класифікує документи в залежності від їх величини схожості відносно запиту. Документ може вважатися знайденим, навіть якщо він лише частково відповідає запиту [14].

1.2.6.3 Імовірнісна модель

Релевантність в цій моделі розглядається як вірогідність того, що даний документ може виявитися цікавим користувачеві. При цьому мається на увазі наявність вже існуючого первинного набору релевантних документів, вибраних користувачем або отриманих автоматично при якому-небудь спрощеному припущенні. Вірогідність виявитися релевантним для кожного наступного документа розраховується на підставі співвідношення знаходження термінів в релевантному наборі до іншого, не релевантного набору документів. Хоча імовірнісні моделі володіють деякою теоретичною перевагою, адже вони розташовують документи в порядку убування, на практиці вони так і не набули великого поширення [14].

Визначення. Для імовірнісної моделі, вага індекс термів змінюється в межах $w_{i,j} \in \{0,1\}$, $w_{i,q} \in \{0,1\}$. Запит q – це підмножина індекс термів. Нехай, R – це множина документів, відомих як релевантні, а $\overline{R}$ - доповнення R (тобто множина не релевантних документів). Нехай $P(R|\overrightarrow{d_j})$ - це імовірність того, що документ dj релевантний до запиту q і $P(\overline{R}|\overrightarrow{d_j})$ - це імовірність того, що dj є не релевантним до запиту q. Релевантність документів dj $sim(dj,q)$ до запиту q визначається як:

$$sim(dj,q) = \frac{P(R|\overrightarrow{d_j}) \times P(R)}{P(\overline{R}|\overrightarrow{d_j}) \times P(\overline{R})} \qquad (1.3)$$

1.2.6.4 Латентно-семантична модель

Жодна з розглянутих вище моделей не має ніякого відношення до семантики текстів. На відміну від них, модель латентно-семантичного індексування може виявити релевантний запиту документ, котрий не буде містити в собі жодного слова з строки пошуку. Цей процес можливий завдяки врахуванні таких особливостей слів, як:

– синонімія – коли одне і теж поняття може описуватися різними словами;

– полісемія – коли одне і теж слово, чи комбінація слів описують різні поняття [18].

ЛСА (латентно-семантичний аналіз) використовує матрицю, котра описує

вживання слів в текстах. Нехай, стовпці матриці відповідають документа, а строки - словам, що зустрічаються в цих документах. Тоді, елементи матриці представляють собою кількість повторень даного слова в документі.

В даній матриці відображаються зв’язки між словами та документами. Під час їх побудови враховується синонімія та полісемія слів.

Математичний метод полягає в застосуванні сингулярного розкладання матриць [17]. Задається матриця A розмірності m*n, де m – число термів у словнику колекції, а n – число документів. Вважається, що $m \geq n$.

Елемент Atd матриці A є вагою терму t у документі d. Тут t є порядковим номером терму у словнику колекції, а d – порядковий номер документа.

Вводиться матриця U розмірності m*m, що складається з ортонормлаьних власних векторів матриці AA^T. Тоді матриця U^TU є одиничною. Аналогічно вводиться матриця V розмірності n*n, що складається з ортонормальних власних векторів матриці A^TA. Відповідно V^TV також є одиничною матрицею.

Діагональними елементами матриці S розмірності n*n є сингулярні числа матриці A – невід'ємні квадратні корені власних чисел матриці A^TA, де $S_{11} \geq S_{22} \geq ...S_{nm}$. Очевидно, що порядок розташування власних векторів матриць AA^T та A^TA відповідає обраному порядку розташування сингулярних чисел. Таким чином, сингулярне розкладання матриці A має вигляд (1.4):

$$A = USV^T \tag{1.4}$$

Вводяться матриця Uk, що є під матрицею матриці U, та утворюється з її перших k стовпців, підматриця Vk^T, що утворюється з перших k стовпців матриці V, а також підматриця Sk, що утворюється з перших k рядків і стовпців матриці S. Отже маємо (1.5):

$$A_k = U_k S_k V_k^T \tag{1.5}$$

На підставі теореми, матриця Ak вважається найближчою до матриці A, ранг якої не перевищує k. Тут відстань між матрицями A і B задається виразом (1.6): [17]

$$\sum_{ij} (A_{ij} - B_{ij})^2 \tag{1.6}$$

Для пошуку велике значення мають матриці Uk та Vk^T. Рядки матриці Uk розглядаються як образи термів у k-мірному речовинному просторі.

Аналогічно. Стовпці матриці розглядаються як образи документів у тому ж k-мірному просторі. Ці вектори задають необхідний опис для термів і документів у k-мірному просторі прихованих латентних факторів [19].

Стовпці матриці V є власними векторами матриці A^TA. Матриця A^TA розглядається як матриця подібності стовпців матриці A, тобто документів. Отже. Стовпці матриці V задають головні напрямки, за якими слід розрізняти документи. Кожний документ подається як крапка в системі координат головних напрямків. Для документа з номером i – це i-рядок матриці V чи стовпець i матриці V^T. У разі використання мало рангової апроксимації матриці A, документа з номером i відповідає стовпець з номером i матриці Vk^T.

Аналогічно, матриця AA^T розглядається як матриць подібності рядків матриці A, тобто термів, що зустрічаються в документах колекції. Стовпці матриці U задають головні напрямки, за якими слід розрізняти терми. Кожний терм подається як крапка в системі координат головних напрямків. Терм з номером j задається рядком j матриці U чи матриці Uk у разі використання мало рангової апроксимації.

Припускається, що семантично близькі терми мають близькі за відстанню образи у просторі латентних факторів. Аналогічно, образи близьких за тематикою в рамках даної колекції документів також мають бути близькими.

Запит q користувача подається як вектор розмірності m, елемент t якого дорівнює 1, якщо терм з номером t входить у запит i-0. Образ запиту q у просторі латентних факторів будується в вигляді (1.7):

$$q = q^T U_k S_k^{-1} \qquad (1.7)$$

Тоді, міра близькості запиту q до документа d оцінюється величиною скалярного добутку векторів q та ${}^T_k[d]$, де ${}^T_k[d]$ позначає стовпець d матриці Vk^T [20] .

1.2.6.5 Переваги та недоліки математичних моделей

В таблиці 1.5 приведені основні переваги та недоліки розглянутих математичних моделей пошуку.

Таблиця 1.5 – Аналіз математичних моделей пошуку

Математична модель	Теоретико-множинна (булева)	Алгебраїчна (векторна)	Алгебраїчна (латентно-семантична)	Імовірнісна
Переваги	- легка в розумінні; - простота в реалізації.	- враховується вага індекс термів; - дозволяє знаходити документи близькі до запиту	- семантичний аналіз тексту; - синонімія та полісемія	документи ранжуються по імовірності бути релевантними.

Продовження таблиці 1.5

Недоліки	не враховується вага індекс термів.	індекс терми, по яким визначається релевантність документу можуть бути взаємно-незалежними.	відсутній морфологічний аналіз термів	- необхідність розділяти початкову множину документів на релевантні та ті що не є такими; - вага індекс термів є бінарною.

1.3 Постановка задачі

Необхідно розробити модуль для системи управління знаннями, який буде відповідати за обробку та зберігання як початкового тексту, так і його індексу і базу даних.

В основних методах індексування, а саме класифікаційному та координатному, є деякі недоліки. В першому, тексти в залежності від їх змісту, відносяться до відповідного класу, в якому накопичуються всі тексти, що мають схожий зміст. Кожному такому тексту відповідає індекс даного класу, що і виступає його пошуковим образом. Недоліком є те, що тексти можуть відрізнятися і при цьому мати один загальний індекс.

Алгоритм координатного індексування документів оснований на врахуванні класифікаційних характеристик присутніх в тексті термінів (слів та словосполучень), характеризуючи ту чи іншу предметну область. Для цього необхідно створення словника термінів предметної області, при чому, в цьому словнику повинні бути установлені зв’язки між термінами та проведена їх класифікація. Такий словник називається тезаурусом. В якості списку ключових слів та словосполучень для тезауруса пропонується використовувати предметний показник спеціалізованої енциклопедії. Вибір конкретної енциклопедії виконує спеціаліст з предметної області, і цей вибір залежить від цілей, які поставлені при створенні тезауруса. В якості дескрипторів (термінів, що являються іменами класів близьких по суті понять) використовуються назви статей енциклопедій, а зв’язаними з ними по суті вважаються слова з предметного показника, що зустрічаються в відповідних статтях.

Основним недоліком координатного індексування, являється створення тезауруса, що потребує досить великих зусиль.

Математична модель латентно-семантичного індексування містить ряд переваг серед інших моделей. Вона враховує семантичні зв’язки слів, а саме: синонімію та полісемію термів тексту.

Враховуючи вищесказане щодо основних механізмів індексування, а також результати аналізу необхідно розробити механізм індексування текстів, який би задовольняв наступні вимоги:

– простота реалізації – від цього буде залежати ціна продукту;

– точність індексування – від точності індексування тексту, залежить якість подальшого його використання в системі управління знаннями;

– смисловий пошук – врахування синонімії, та словоформ термів;

– відкрита архітектура – розширення можливостей модулю без затрати на це великих ресурсів;

– багатомовність – мається на увазі незалежність роботи механізму від вибраної мови;

– висока швидкість пошуку;

– висока швидкість оброблення текстових документів;

– простота інтерфейсу користувача.

2 ОПИС РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

В даному розділі описується система дворівневого індексування текстів. Для системи представлена модель у вигляді діаграм використання, класів, та послідовності.

2.1 Аналіз вимог

Перед тим як почати проектувати систему, необхідно визначити всі процеси, які необхідні для вирішення поставленої задачі.

Діаграма варіантів використання описує можливі варіанти роботи з системою. Діаграма використання для системи створення індексів зображена на рисунку 2.1.

Перед тим як відкривати текст для його індексування, в систему необхідно додати певні дані. Цими даними являються словники: 1) словник службових слів – використовується для очищення поточного текстового документу від службових слів; 2) морфологічний словник – система використовує його для морфологічного аналізу тексту, очищеного від службових слів; 3) словники тематик – з їх допомогою закладені в систему алгоритми визначають предметну область документу.

Далі відкриваємо текстовий документ, який бажаємо проаналізувати. Після відкриття, він обробляється системою: з тексту видаляються службові слова, проводиться морфологічний аналіз, та визначається тематика тексту. І вже оброблений текст зберігається в БД (базу даних).

Після цих дій, адміністратору необхідно створити індекс поточного тексту. За це відповідають варіанти «Індекс І-го рівня» та «Індекс ІІ-го рівня». Для створення індексу, відбувається звернення до БД і потім вже індекс записується в текстовий файл з відповідним ім’ям документу (таким як в тексту). Ім’я папки, в котру зберігається індекс, має назву відповідної предметної області.

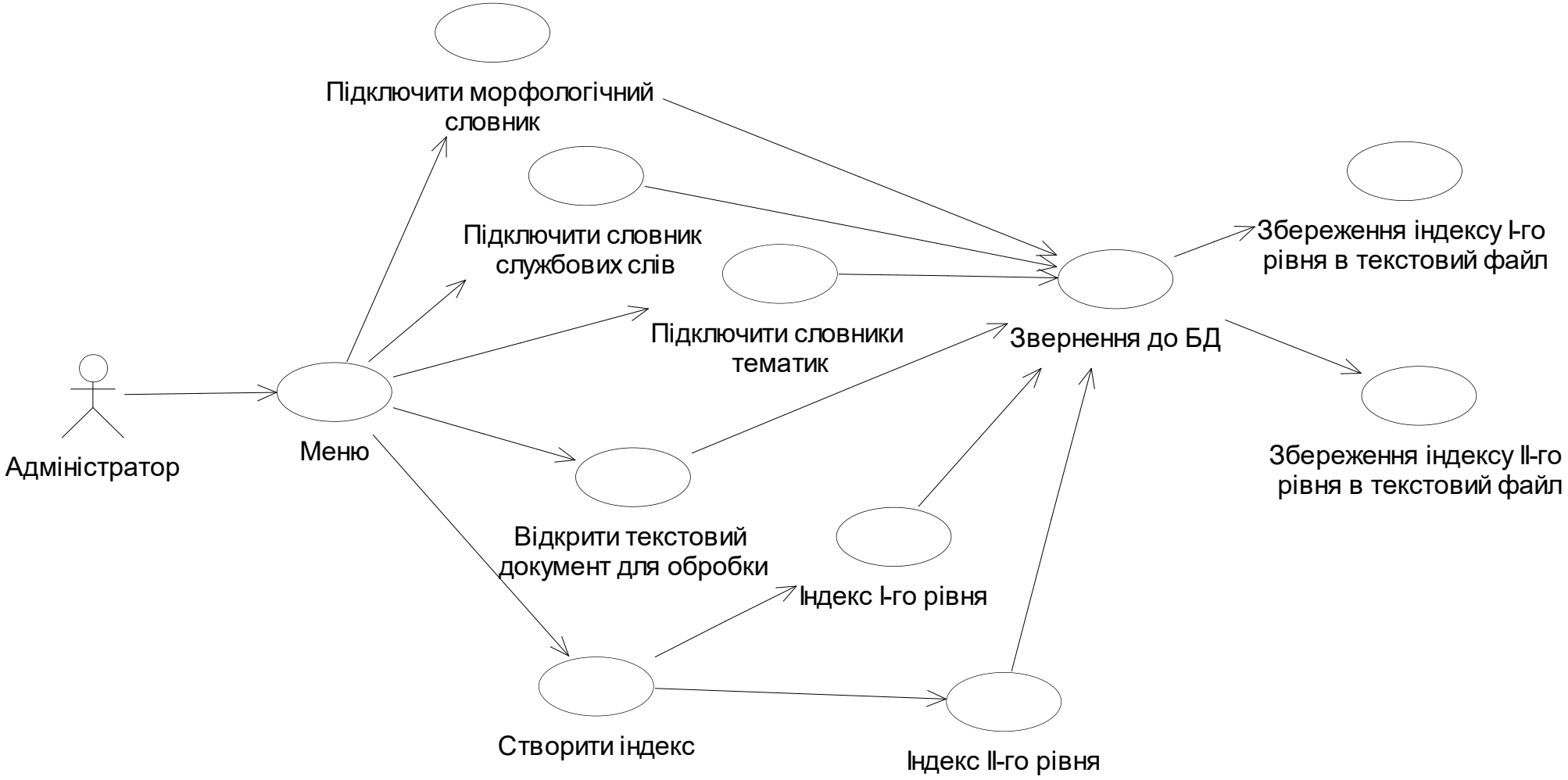


Рисунок 2.1 – Діаграма варіантів використання системи створення індексу текстових документів

Для перевірки точності створення індексу, розроблена пошукова частина програмного продукту. Діаграма використання для пошуку зображена на рисунку 2.2.

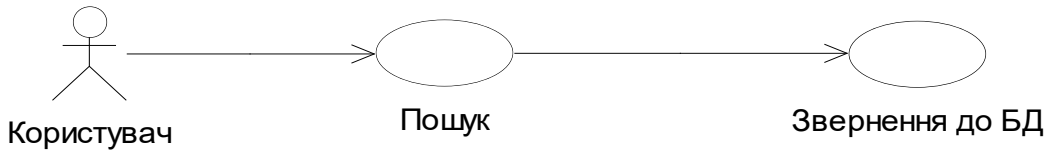


Рисунок 2.2 – Діаграма варіантів використання пошуку

Як видно з діаграми, користувачу пропонується виконати пошук. Для цього необхідно ввести строку пошуку, в результаті якого відбувається звернення до БД, де відбувається морфологічний аналіз запиту, та його предметна область. Відповідно до визначеної предметної області запиту, аналізуються індекси, що знаходяться в папці під назвою тематики. В результаті, на екран виводяться назви документів, в яких присутні слова, чи словосполучення запиту.

2.2 Концептуальна модель системи

При розробці механізму індексування необхідно створити модель, котра буде відображати всі процеси предметної області пошуку: від потрапляння документу в систему, його індексування до оброблення запиту на пошук, та його виконання. На рисунку 2.3 зображена діаграма класів концептуальної моделі, в основу якої закладений механізм обробки (індексування) текстових документів.

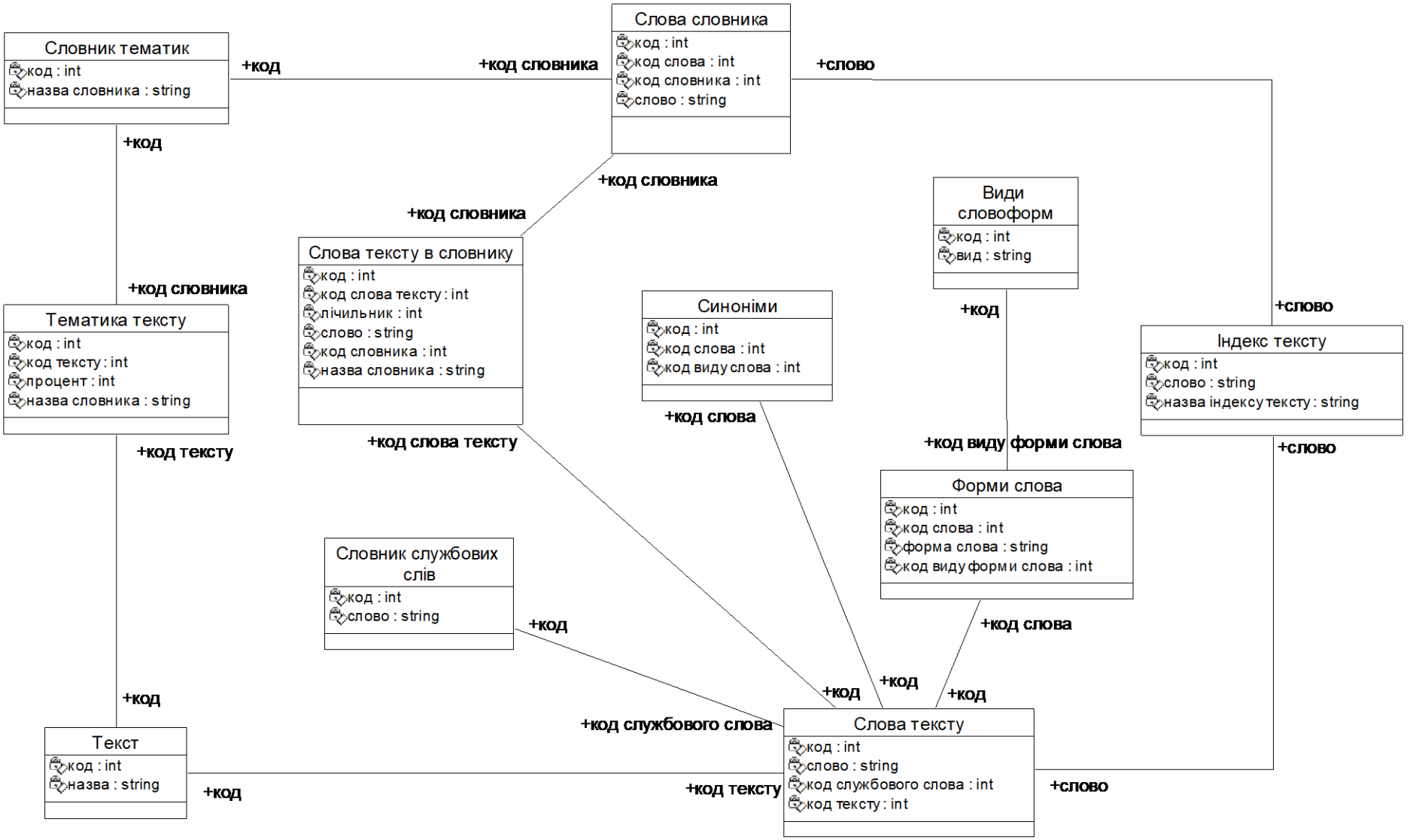


Рисунок 2.3 – Концептуальна модель системи індексування текстів.

Опишемо призначення кожного класу:

«Текст»: оброблює всі назви текстових документів, що знаходяться в базі даних (Рисунок 2.4).

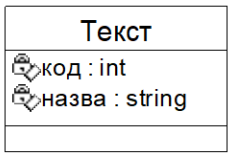


Рисунок 2.4 – Клас «Текст»

«Слова тексту»: даний клас оброблює всі слова тексту, котрі пройшли фільтрацію від службових слів (Рисунок 2.5).

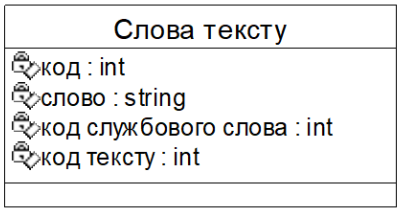


Рисунок 2.5 – Клас «Слова тексту»

«Словник тематик»: являє собою атрибути та методи необхідні для підключення в систему словників предметних областей (Рисунок 2.6).

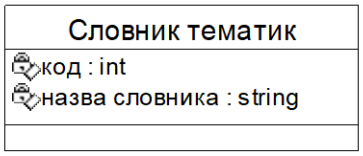


Рисунок 2.6 – Клас «Словник тематик»

«Слова словника»: клас призначений для обробки слів з усіх словників предметних областей (Рисунок 2.7).

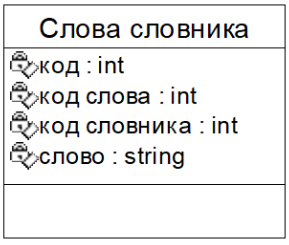


Рисунок 2.7 – Клас «Слова словника»

«Словник службових слів»: являє собою атрибути та методи необхідні для підключення в систему словника службових слів (Рисунок 2.8).

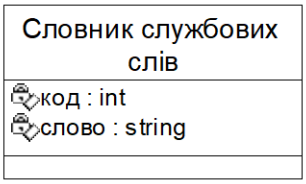


Рисунок 2.8 – Клас «Словник службових слів»

«Форми слова», «Види словоформ» та «Синоніми»: призначення перших двох класів, оброблювати всі словоформи конкретного слова і приводити дане слово до простої форми (Рисунок 2.9 та 2.10). «Синоніми» - виконує аналіз слів однакових по суті, але різних по звучанню слів (Рисунок 2.11). В основу реалізації даного класу покладені основна модифікована латентно-семантична модель індексування текстів. Сама ж модель описана в розділі аналізу існуючих рішень.

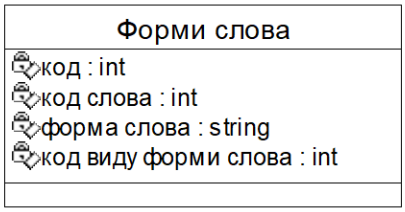


Рисунок 2.9 – Клас «Форми слова»

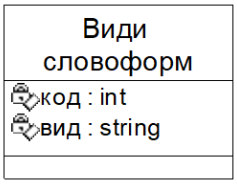


Рисунок 2.10 – Клас «Види словоформ»

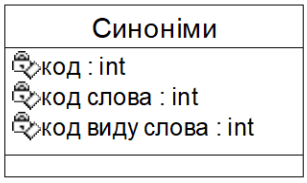


Рисунок 2.11 – Клас «Синоніми»

«Слова тексту в словнику»: даний клас оброблює слова поточного тексту, котрі мають зв'язок з словами словників. Будується базис тексту, тобто яке слово, скільки раз зустрічається в тексті, і в якому словнику воно присутнє (Рисунок 2.12).

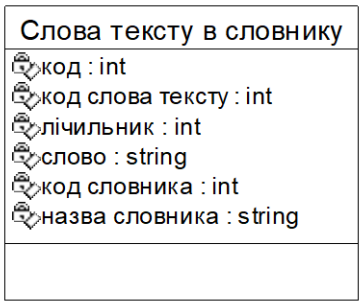


Рисунок 2.12 – Клас «Слова тексту в словнику»

«Тематика тексту»: на основі даних отриманих від попередньо описаного класу, відбувається визначення процентного співвідношення тексту до кожної з предметних областей в словниках яких зустрічалися слова поточного тексту (Рисунок 2.13).

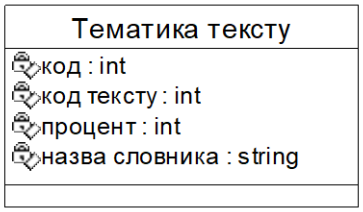


Рисунок 2.13 – Клас «Тематика тексту»

«Індекс тексту»: даний клас являє собою кінцівку обробки початкового текстового документу. На основі даних отриманих від всіх попередніх класів, будується індекс тексту і зберігається в текстовому документі в папці, відповідно до предметної області (Рисунок 2.14).

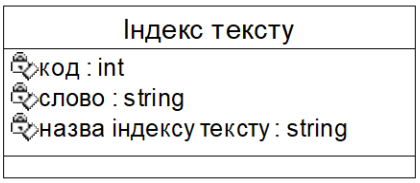


Рисунок 2.14 – Клас «Індекс тексту»

2.3 Структура системи

На рисунку 2.15 показана структура системи індексування текстів. З рисунку видно, що вона складається з підсистем, котрі виконують певні функції.

Підсистема обробки початкового тексту: коли текст попадає в систему, в ньому присутні безліч службових слів, котрі не несуть в собі ніякої суті, також в тексті зустрічається безліч однакових слів, але з різними закінченнями.

Спочатку, з поточного тексту видаляються службові слова, котрі не несуть в собі інформації. Далі необхідно провести морфологічний аналіз. В результаті останнього слова тексту приводяться до простої форми. Слова, що мали різні закінчення, але однакову основу і вважалися різними, після морфологічного аналізу вважаються рівними.

Містить в собі: модуль, що відповідає за фільтр службових слів, модуль морфологічного аналізу, та визначення синонімії. Дана підсистема основана на аспектах латентно-семантичного аналізу, та вдосконалена за рахунок врахування словоформ термів. Це дає можливість виділити семантику слів текстів.

Підсистема визначення тематики тексту: перед тим, як приступити до побудови індексу тексту, необхідно визначити його тематику. Виконавши останнє, порівнюючи слова тексту, з словами словника визначеної предметної області можна приступати до створення індексу. В даній підсистемі присутні: модуль підключення словників тематик та функції для її визначення.

Підсистема побудови індексу: отримавши оброблений текст, та визначивши його тематику можна приступати до побудови індексу. Дана підсистема містить в собі наступні модулі: обробки словника визначеної тематики тексту, котрий в свою чергу взаємодіє з модулем, що являє собою функції для роботи зі словами тексту.

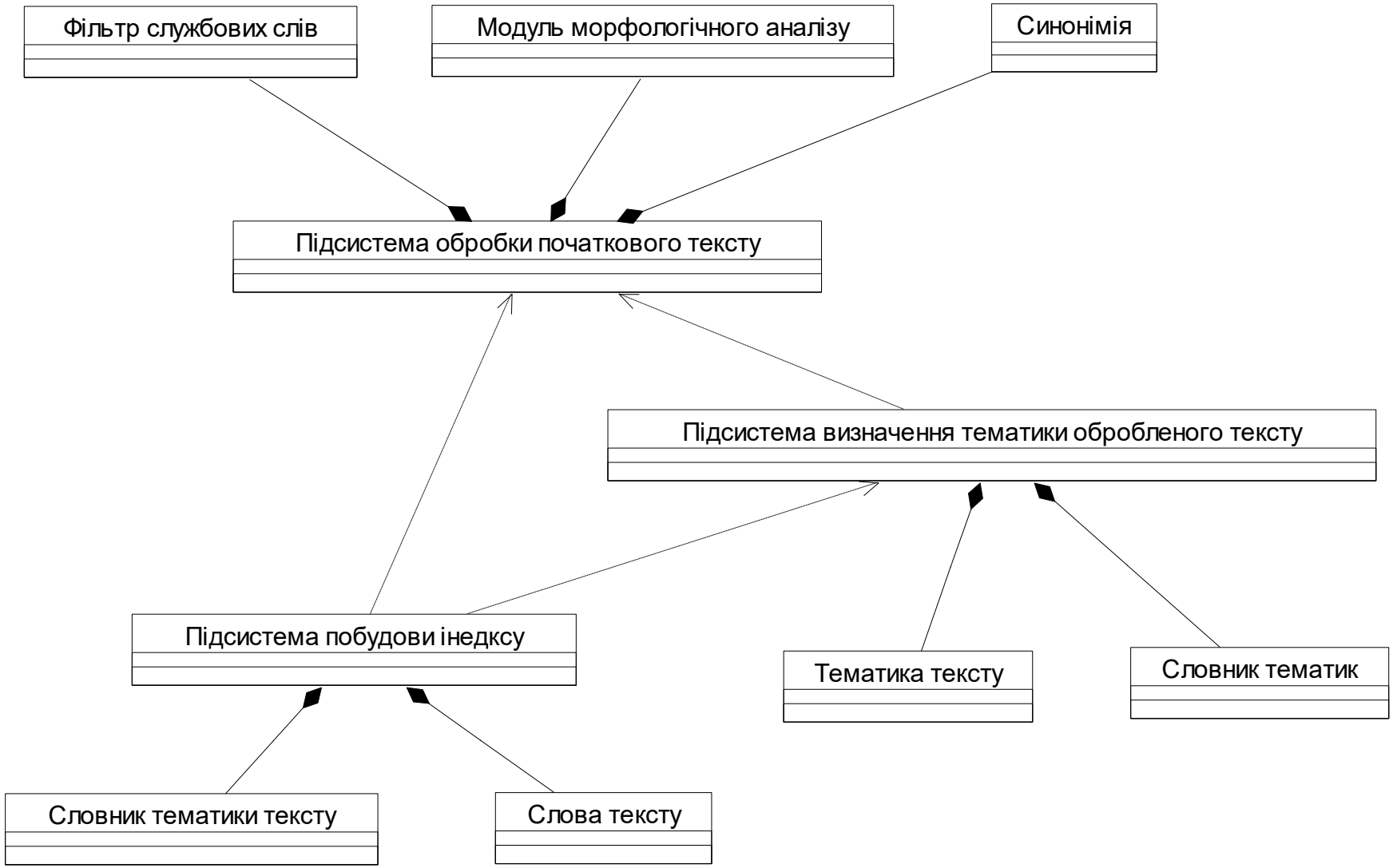


Рисунок 2.15 – Структура системи індексування текстів

2.4 Алгоритм роботи системи індексування текстів

Алгоритм розробленої системи являється результатом схрещення двох механізмів індексування: класифікаційного та координатного. В першому, тексти в залежності від їх змісту, відносяться до відповідного класу, в якому накопичуються всі тексти, що мають схожий зміст. Кожному такому тексту відповідає індекс даного класу, що і виступає його пошуковим образом. Недоліком є те, що тексти можуть відрізнятися і при цьому мати один загальний індекс.

Отже, недоліками класифікаційного індексування є:

– низький рівень точності індексу;

– присвоєння різним текстам спільного індексу.

Переваги класифікаційного індексування - це:

– компактність індексів (індекс створюється для класу, а не для кожного тексту індивідуально);

– велика швидкість пошуку (оскільки індекс один на клас, і слова запиту є в цьому індексі результатом будуть всі тексти даного класу).

Алгоритм координатного індексування документів оснований на врахуванні класифікаційних характеристик присутніх в тексті термінів (слів та словосполучень), характеризуючи ту чи іншу предметну область. Для цього необхідно створення словника термінів предметної області, при чому, в цьому словнику повинні бути установлені зв’язки між термінами та проведена їх класифікація. Такий словник називається тезаурусом. Його створення потребує великих зусиль програміста, що і є недоліком цього алгоритму.

Тобто, недоліками координатного індексування є:

– мала швидкість пошуку;

– складність врахування зв’язків між словами в тезаурусі.

Переваги координатного індексування:

– велика точність індексу;

– можливість створення чіткої системи підказок під час пошуку.

2.4.1 Алгоритм підготовки системи індексування текстів

Для того, щоб система могла будувати індекси текстів, необхідно підготувати її до цього. А саме, підключити необхідні словники: словники тематик, службових слів та морфологічний словник. Алгоритм підготовки системи до індексування зображено на рисунку 2.16.

Рисунок 2.16 – Алгоритм підготовки системи для подальшого індексування текстів.

Опишемо кожен крок алгоритму зображеного на рисунку 2.16:

Крок 1 Підключити словник службових слів. Атрибути та методи

обробки словника службових слів описані в класі «Словник службових слів» (Рисунок 2.8).

Даний словник використовується в системі для очищення тексту від службових слів. Алгоритм запису словника до БД описано на рисунку 2.17.

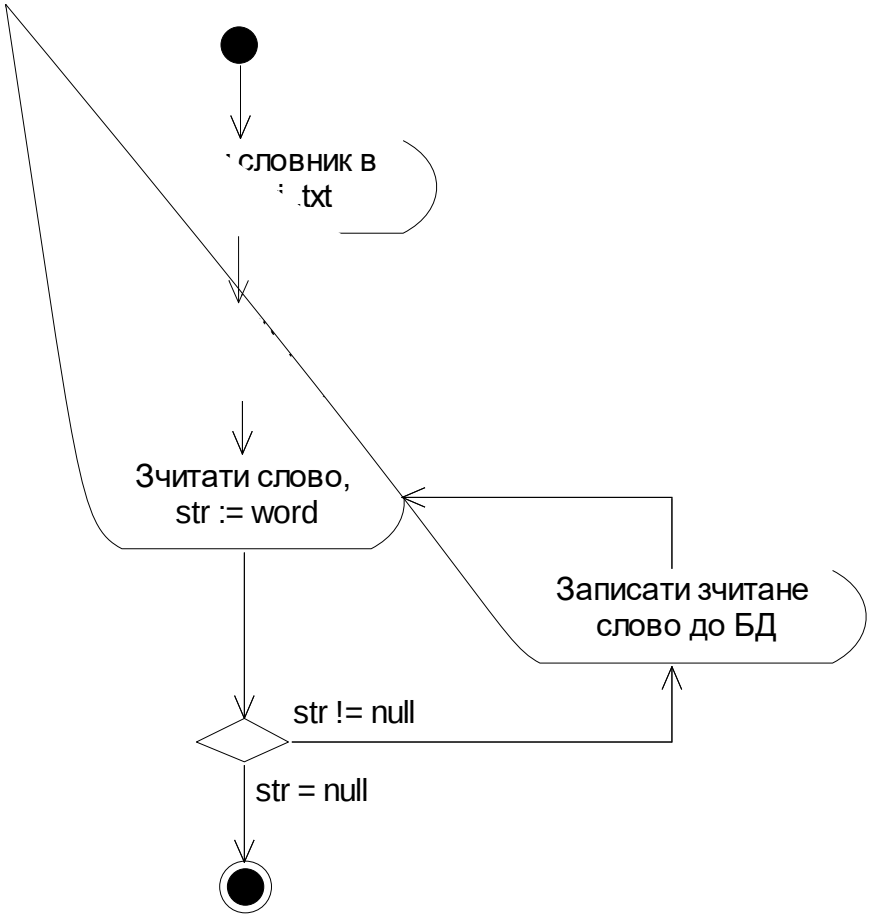


Рисунок 2.17 – Алгоритм підключення словника службових слів в систему індексування текстів

Крок 2 Підключити словники тематик. Дані словники використовуються для визначення тематики поточного тексту, тематики пошукового запиту а також для побудови індексу. Алгоритм підключення словників тематик зображено на рисунку 2.18. На даному етапі створюється три таблиці: 1) таблиця назв словників, де вказано назву словника та його ідентифікатор; 2) таблиця слів, де вказується слово, та його ідентифікатор; 3) таблиця слів в словниках з полями: 1) слово; 2) його

ідентифікатор; 3) ідентифікатор словника в якому дане слово присутнє. Атрибути та методи обробки словників тематик описані в класах «Словник тематик» (2.6) та «Слова словника» (2.7).

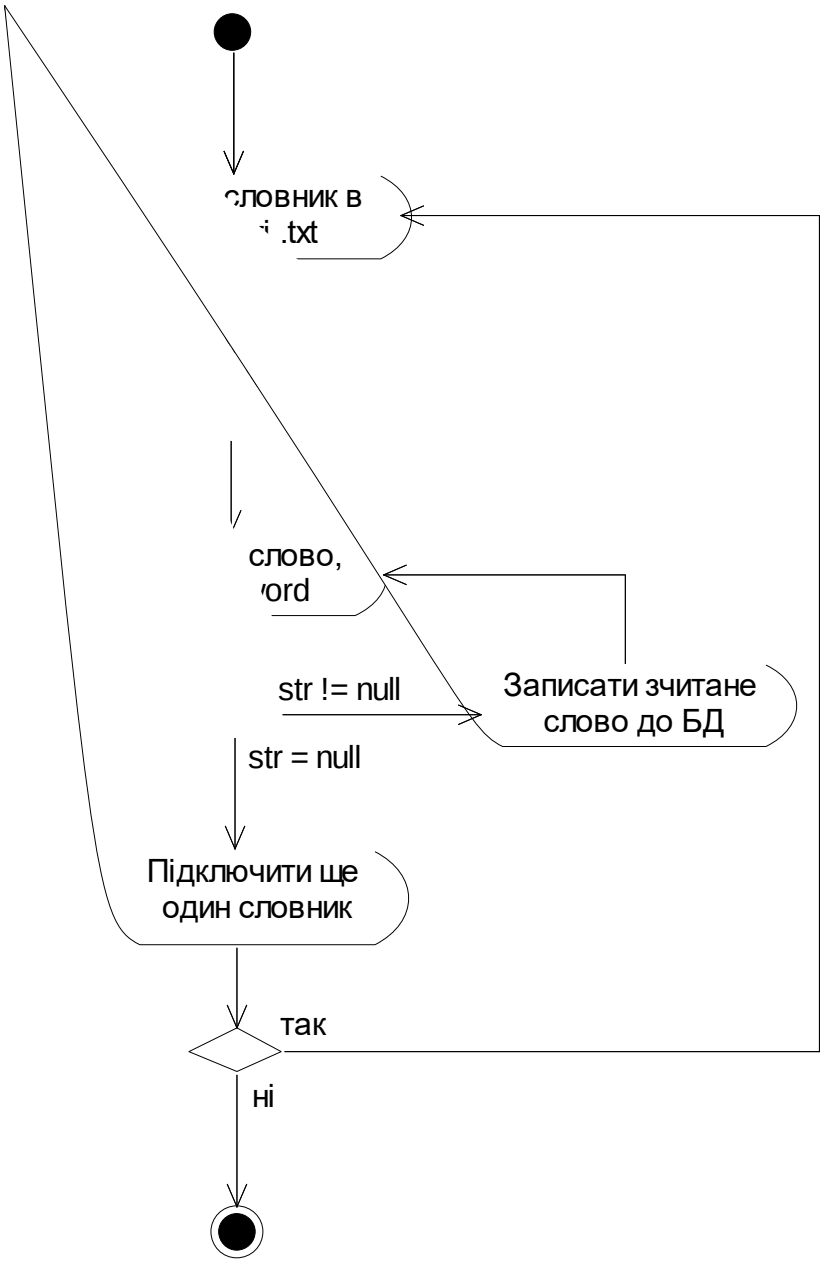


Рисунок 2.18 – Алгоритм підключення словників тематик до БД в системі індексування текстів.

Крок 3 Підключити морфологічний словник. Морфологічний словник використовується для приведення однакових слів до однієї загальної форми. Коли в систему потрапляє текст для створення індексу, відбувається його очищення від службових слів та морфологічний

аналіз, і до БД записуються всі слова в простій формі. При чому, слова записані в тій послідовності в якій вони знаходяться в текстів. В результаті отримаємо таблицю з полями: 1) ідентифікатор; 2) слово тексту; 3) ідентифікатор тексту.

Вище були описані класи, що містять в собі необхідні методи та атрибути для виконання даного кроку: «Форми слова» (Рисунок 2.9) та «Види словоформ» (Рисунок 2.10).

2.4.2 Алгоритм створення індексу І-го рівня

Алгоритм створення індексу І-го рівня зображено на рисунку 2.19. Результатом алгоритму являється створення текстового файлу з розширенням txt.

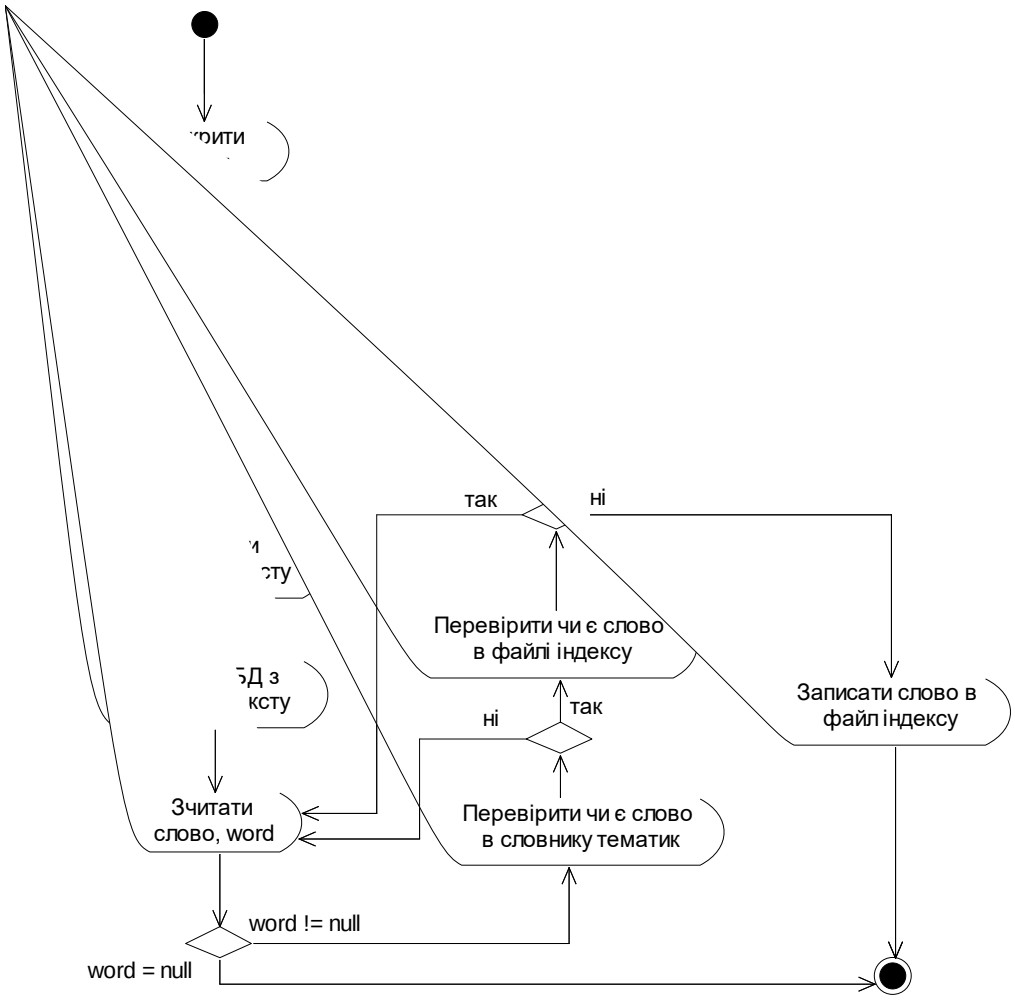


Рисунок 2.19 – Алгоритм створення індексу І-го рівня

Опишемо кожен крок алгоритму:

Крок 1 Відкрити текст. В діалоговому вікні, користувачу необхідно вибрати текстовий файл і натиснути «ОК». В результаті, викликаються атрибути та методи класу «Текст» (Рисунок 2.4) та класу «Слова тексту» (Рисунок 2.5).

Крок 2 Очистити текст від службових слів. В результаті виконання даного кроку, текст очищується від службових слів. Алгоритм фільтрування тексту зображено на рисунку 2.20. Клас, що містить в собі необхідні методи, називається «Словник службових слів» (Рисунок 2.8).

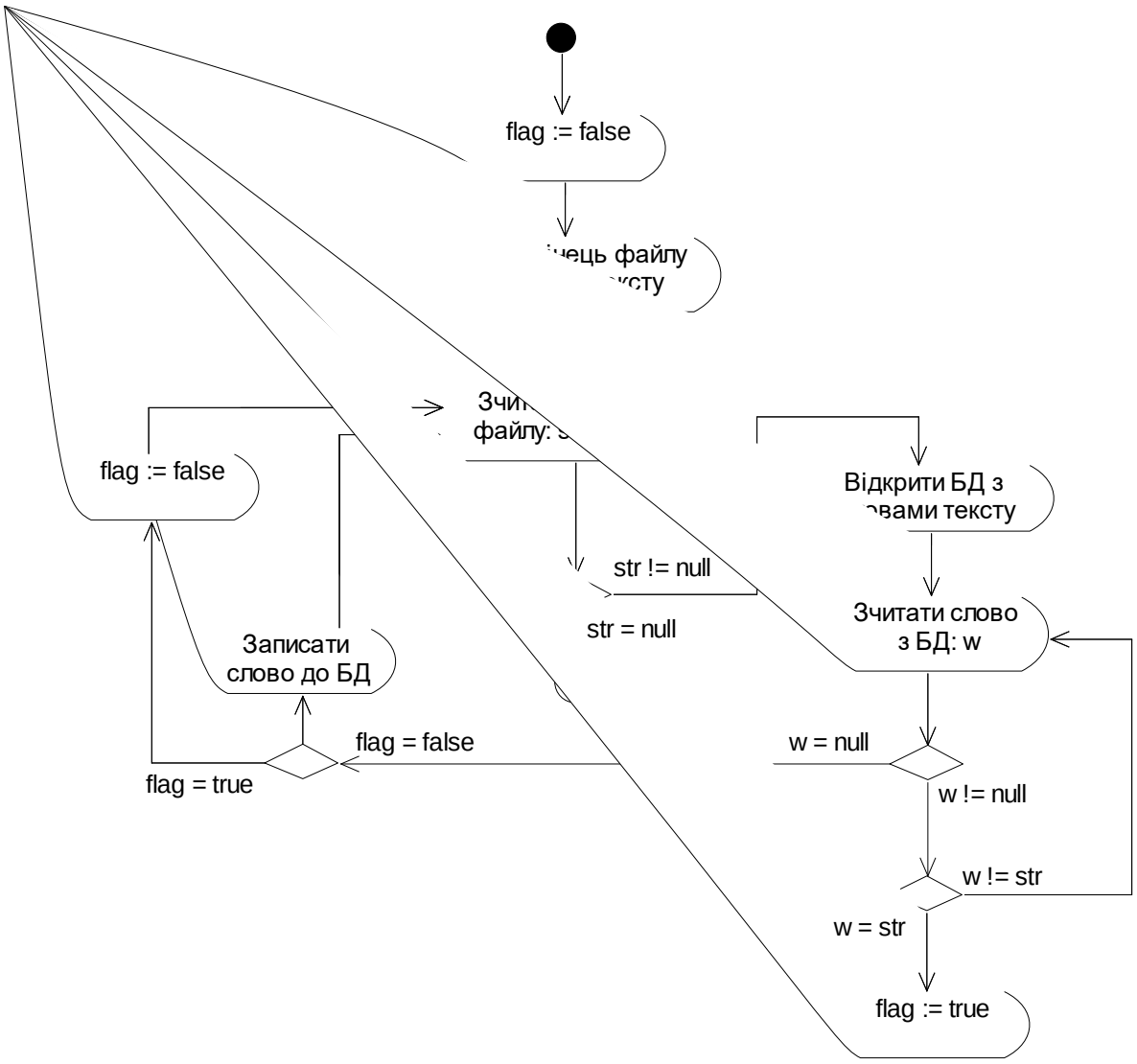


Рисунок 2.20 – Алгоритм очищення поточного тексту від службових слів в системі індексування текстів

Крок 3 Морфологічний аналіз тексту. Проводиться з метою приведення всіх

слів тексту до простої форми. Алгоритм морфологічного аналізу зображено на рисунку 2.21. Методи реалізації кроку описані в класах: «Форми слова» (Рисунок 2.9) та «Види словоформ» (Рисунок 2.10).

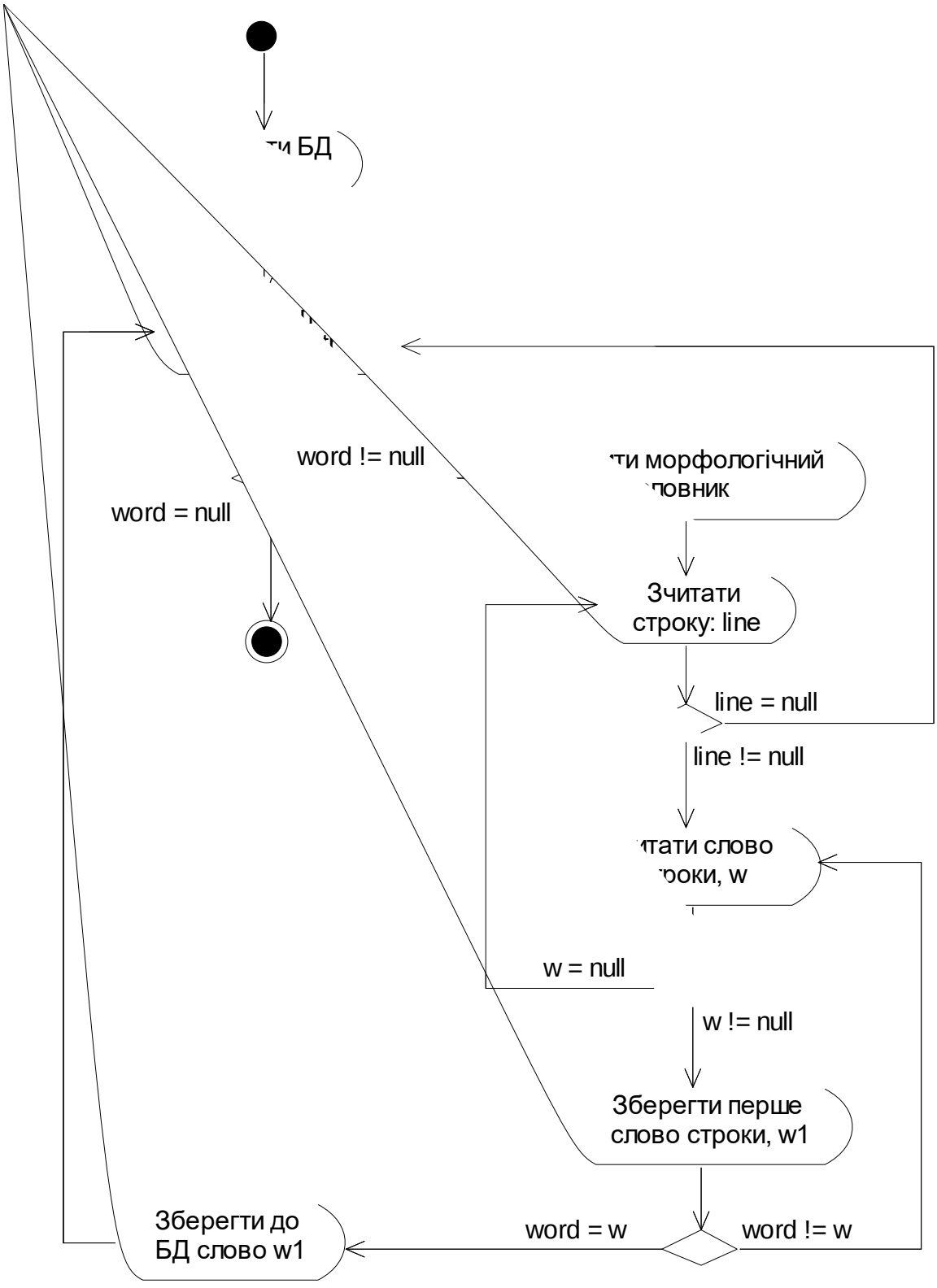


Рисунок 2.21 – Алгоритм морфологічного аналізу поточного тексту

Корок 4 Визначити тематику тексту. Алгоритм визначення тематики тексту зображено на рисунку 2.22. Для визначення тематики створюється таблиця в БД з наступними полями: 1) ідентифікатор слова тексту; 2) ідентифікатор поточного тексту; 3) скільки разів в тексті присутнє; 4) слово; 5) словник в якому воно присутнє; 6) ідентифікатор словника.

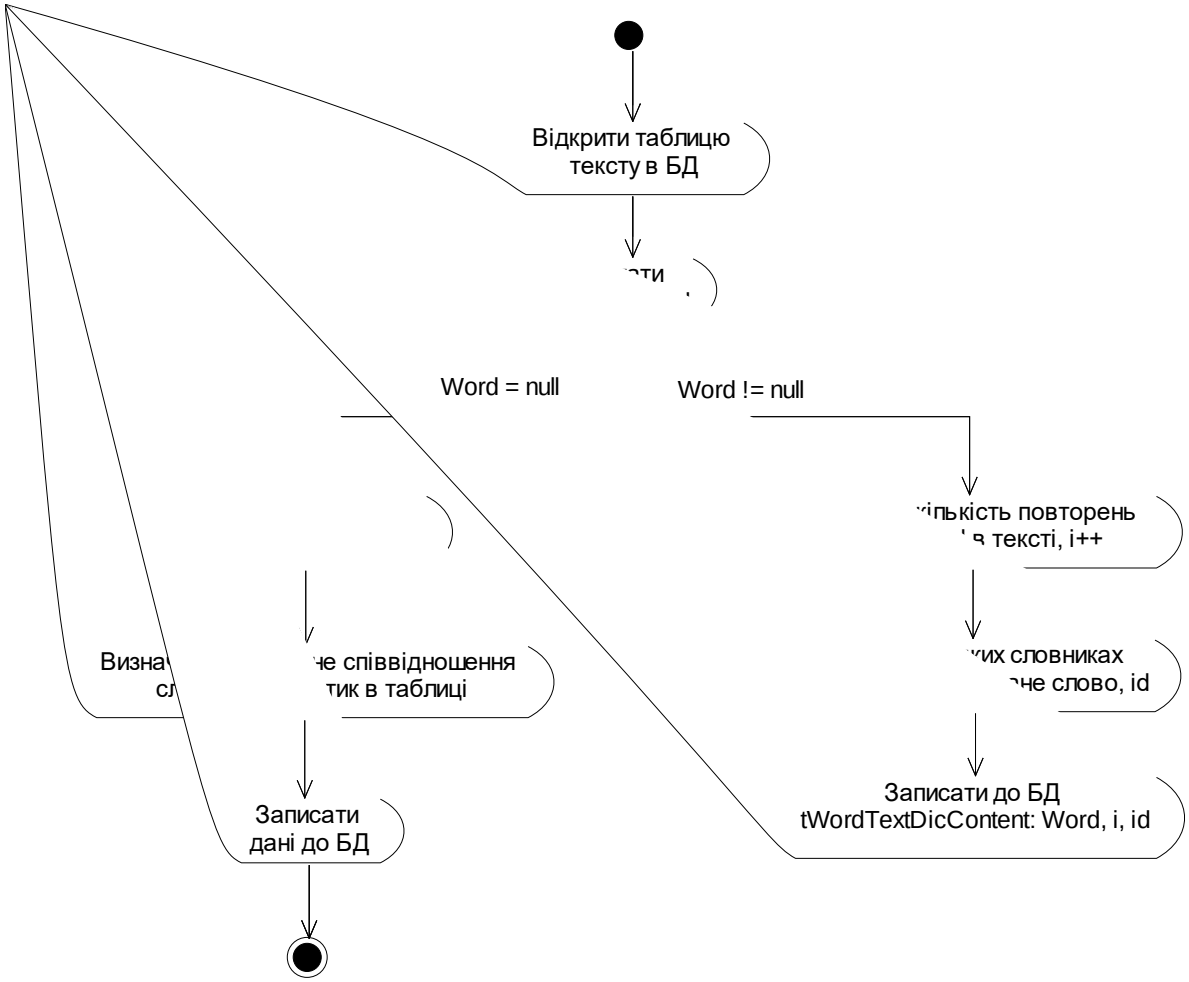


Рисунок 2.22 – Алгоритм визначення тематики тексту в системі індексування

Для реалізації даного алгоритму використовуються методи класів: «Слова тексту в словнику» (Рисунок 2.12) та «Тематика тексту» (Рисунок 2.13).

Крок 5-12 Після очищення тексту від службових слів, морфологічного аналізу, та визначення тематики аналізуємо кожне його слово. Перебираємо послідовно всі слова обробленого тексту і перевіряємо їх

наявність в словнику визначеної тематики. Якщо слово присутнє в словнику і відсутнє в файлі індексу, то воно зберігається в текстовий файл, який і являється індексом І-го рівня. Використовуються атрибути та методи класу «Індекс тексту» (Рисунок 2.14).

2.4.3 Алгоритм створення індексу ІІ-го рівня

На відміну від алгоритму І-го рівня, де індексами являються окремі слова, індекс ІІ-го рівня складається з словосполучень тексту. Алгоритм створення індексу зображено на рисунку 2.23.

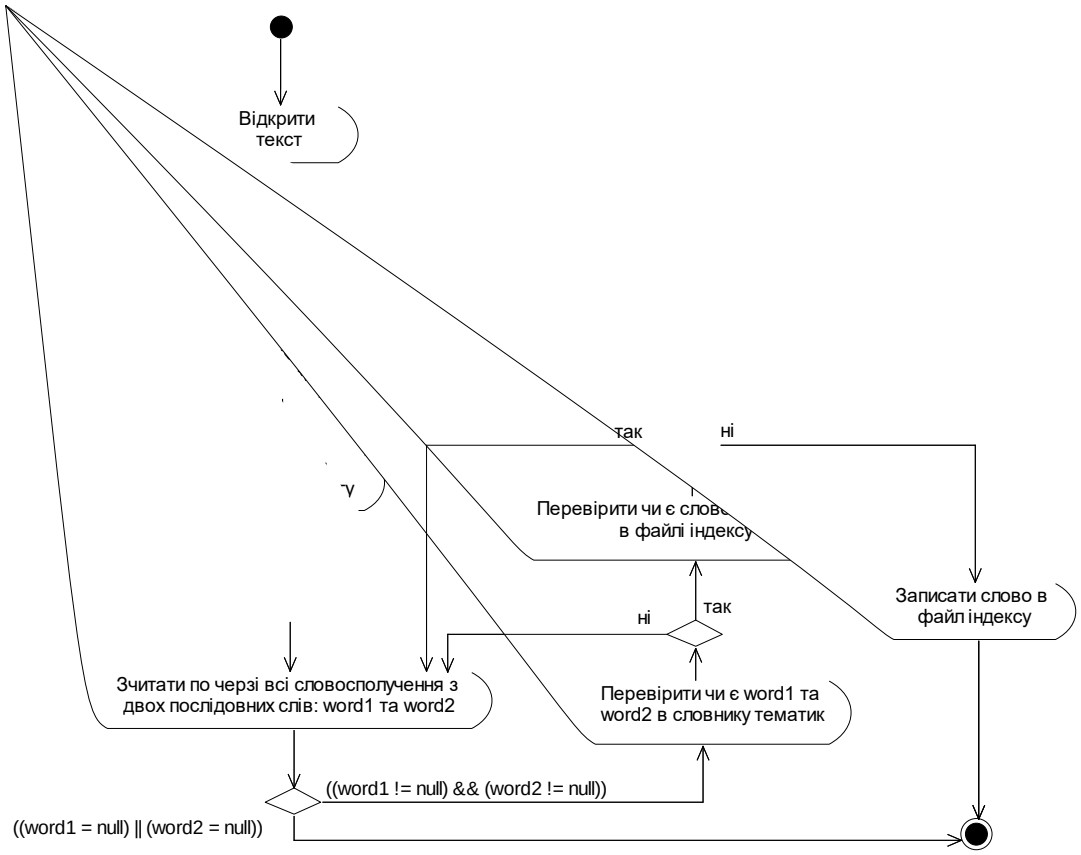


Рисунок 2.23 – Алгоритм побудови індексу ІІ-го рівня

Крок 1 Відкрити текст. В діалоговому вікні, користувачу необхідно вибрати текстовий файл і натиснути «ОК». В результаті, викликаються атрибути та методи класу «Текст» (Рисунок 2.4) та класу «Слова тексту» (Рисунок 2.5).

Крок 2 Очистити текст від службових слів. В результаті виконання даного кроку, текст очищується від службових слів. Алгоритм фільтрування тексту зображено на рисунку 2.20. Клас, що містить в собі необхідні методи, називається «Словник службових слів» (Рисунок 2.8).

Крок 3 Морфологічний аналіз тексту. Проводиться з метою приведення всіх слів тексту до простої форми. Алгоритм морфологічного аналізу зображено на рисунку 2.21. Методи реалізації кроку описані в класах: «Форми слова» (Рисунок 2.9) та «Види словоформ» (Рисунок 2.10).

Корок 4 Визначити тематику тексту. Для реалізації даного кроку використовуються методи класів: «Слова тексту в словнику» (Рисунок 2.12) та «Тематика тексту» (Рисунок 2.13). Алгоритм визначення тематики тексту зображено на рисунку 2.22. Для визначення тематики створюється таблиця в БД з наступними полями : 1) ідентифікатор слова тексту; 2) ідентифікатор поточного тексту; 3) скільки разів в тексті присутнє; 4) слово; 5) словник в якому воно присутнє; 6) ідентифікатор словника.

Попередні чотири кроки аналогічні алгоритму створення індексу І-го рівня.

Крок 5-12 Після очищення тексту від службових слів, морфологічного аналізу, та визначення тематики аналізуємо кожне його словосполучення, слова якого знаходяться в тексті послідовно одне за одним. Перебираємо такі словосполучення обробленого тексту і перевіряємо наявність його слів в словнику визначеної тематики. Якщо слова присутні в словнику і дане словосполучення відсутнє в файлі індексу, то воно зберігається в текстовий файл, який і являється індексом ІІ-го рівня. Використовуються атрибути та методи класу «Індекс тексту» (Рисунок 2.14).

2.4.4 Алгоритм пошуку строки запиту в індексах

Для перевірки якості індексування текстів, було розроблено пошуковий модуль системи. Алгоритм пошуку текстів по індексах зображено на рисунку 2.24.

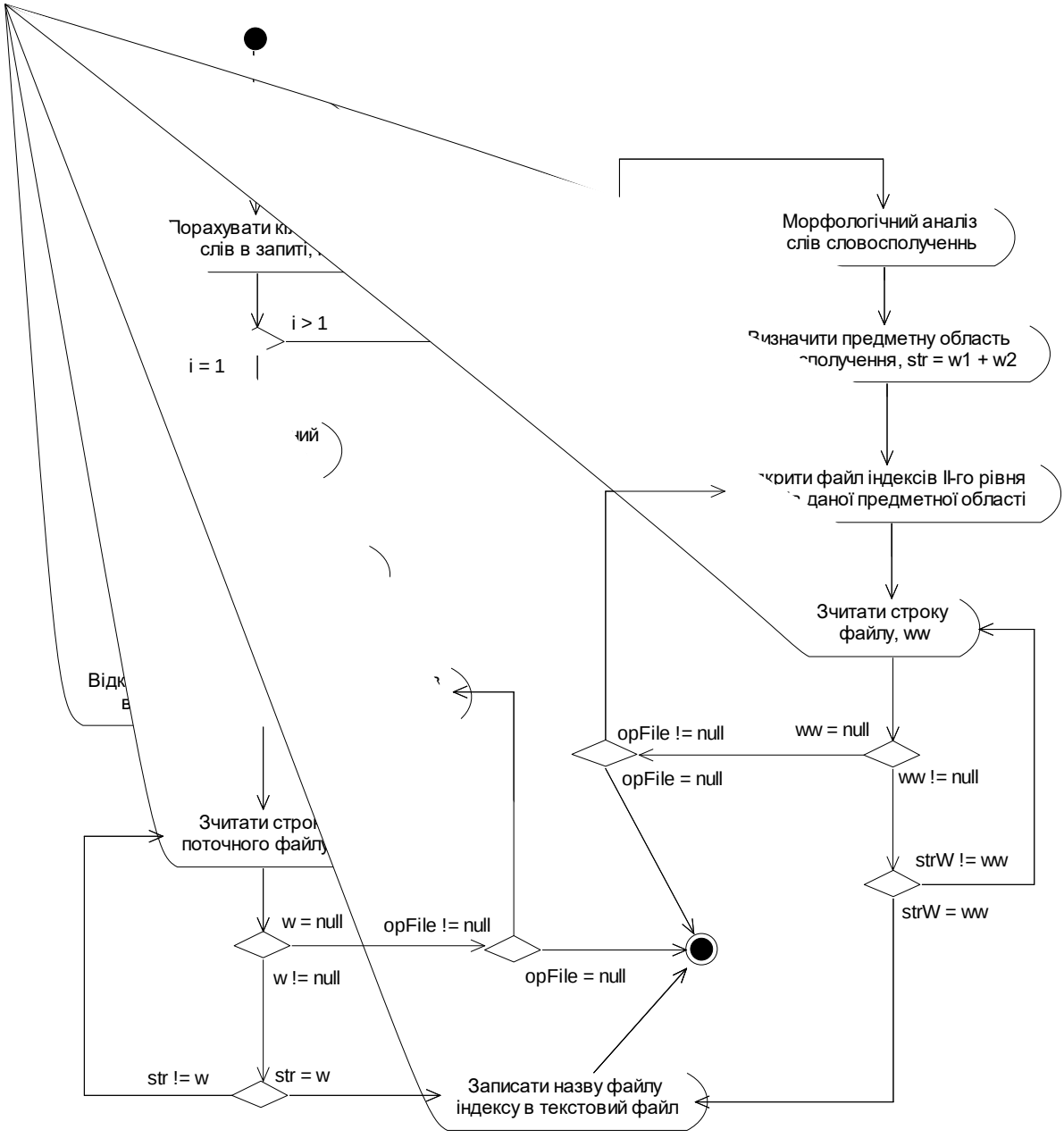


Рисунок 2.24 – Алгоритм пошуку строки запиту в індексах I-го та II-го рівнів

Крок 1 Зчитати строку запиту. Система зчитує строку запиту введену

користувачем. Клас, що містить с собі атрибути та методи обробки строки запиту зображено на рисунку 2.25.

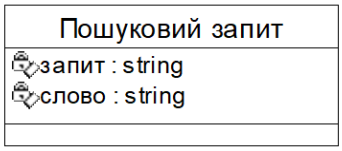


Рисунок 2.25 – Клас «Пошуковий запит»

Крок 2 Необхідно перевірити скільки слів було в запиті. Якщо одне слово, то будуть аналізуватися індекси I-го рівня (Крок 4). Якщо більше одного слова, тобто, словосполучення, то аналізу підлягають індекси II-го рівня (Крок 11). Використовуються методи класу «Пошуковий запит».

Крок 4 Морфологічний аналіз слова. Оскільки в індексі всі слова зберігаються в простій формі, то строку запиту система приводить до простої форми і вже оперує далі з цим словом. Алгоритм морфологічного аналізу зображено на рисунку 2.26. Методи морфологічного аналізу слів запиту аналогічні методам аналізу слів тексту, описані в класах «Форми слова» (Рисунок 2.9) та «Види словоформ» (Рисунок 2.10).

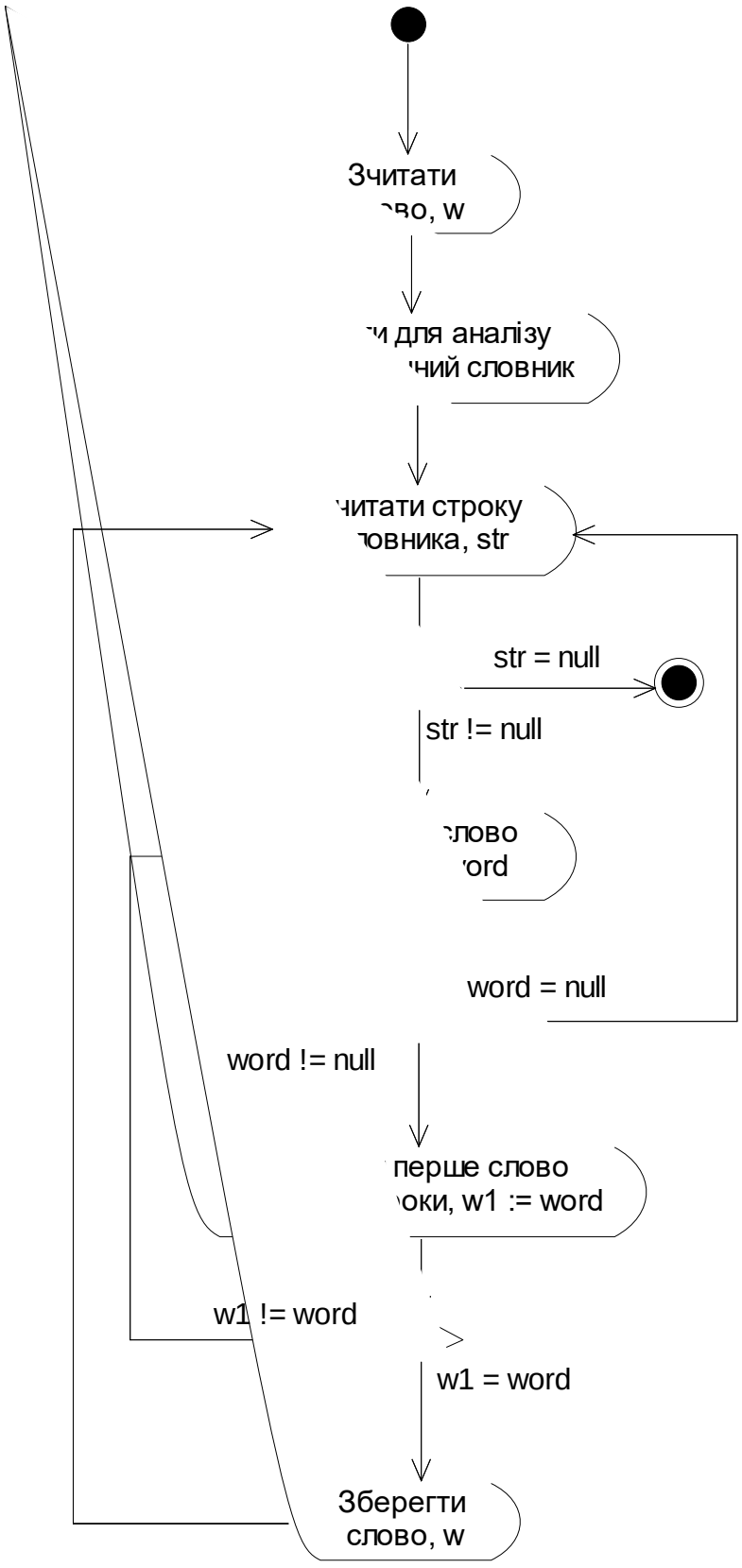


Рисунок 2.26 – Алгоритм морфологічного аналізу строки запиту

Крок 5 Визначити предметну область слова. Слово може належати більш ніж

до однієї предметної області. Алгоритм визначення предметної області слова зображено на рисунку 2.27.

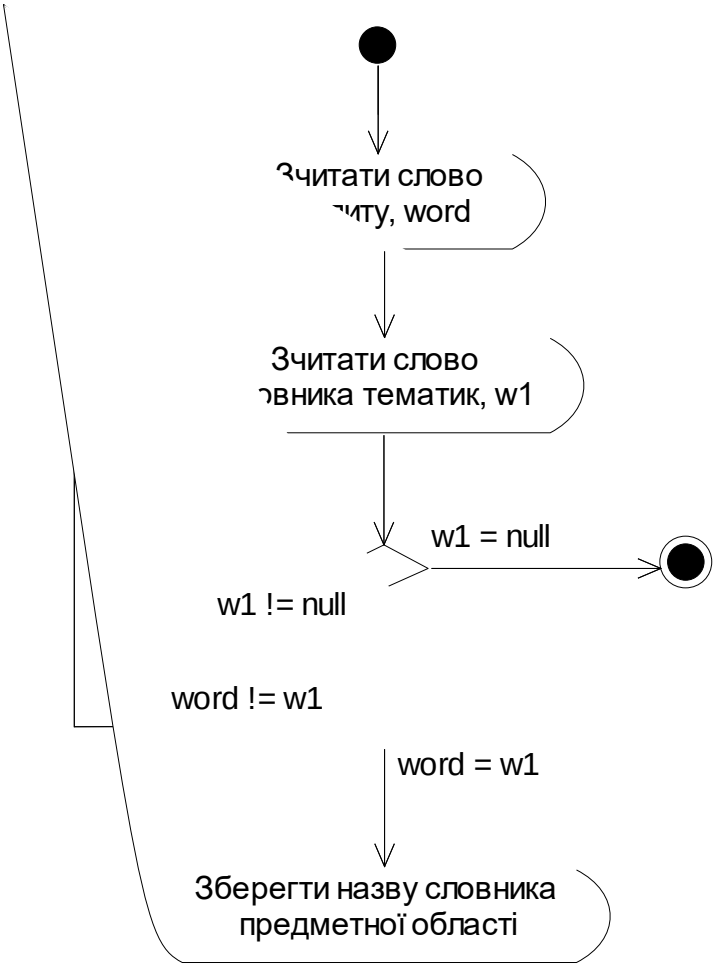


Рисунок 2.27 – Алгоритм визначення предметної області слова строки запиту

Для реалізації даного кроку використовуються методи класу: «Тематика тексту» (Рисунок 2.13).

Крок 5 - 10 Аналізуємо індекси І-го рівня текстів, що належать предметним областям визначеним на попередньому кроці. Якщо в індексі присутнє слово запиту, то назва файлу індексу записується в текстовий файл (Крок 18). Використовуються атрибути та методи класу «Індекс тексту» (Рисунок 2.14).

Крок 11 Морфологічний аналіз словосполучення. Якщо в запиті більше одного слова, то послідовно перебираються всі словосполучення і проводиться морфологічний аналіз кожного слова аналогічно Кроку 3.

Використовуються методи тих самих класів, про котрі йшла мова в Кроку 4 даного алгоритму.

Крок 12 Визначити предметну область словосполучення. Алгоритм визначення предметної області словосполучення аналогічний Кроку 4. Якщо два слова словосполучення належать одній предметній області, то і словосполучення належить даній предметній області.

Крок 13-17 Перебираємо словосполучення строки запиту, і аналізуємо всі індекси II-го рівня тих предметних областей, до яких належить дане словосполучення. Якщо в індексі знаходяться обидва слова словосполучення, то назва файлу індексу зберігається в текстовий файл (Крок 18).

Крок 18 Записати назву файлу індексу в текстовий документ. Даний документ тримає в собі назви текстів, в яких було виділено строку запиту. Для реалізації Кроків 13-18 використовуються атрибути та методи класу «Індекс тексту» (Рисунок 2.14).

2.5 Діаграми послідовності системи

Діаграма послідовності показує послідовність виконання операцій при роботі з системою. Так, наприклад, діаграма послідовності підготовки системи для подальшого її використання з ціллю індексування має вигляд зображений на рисунку 2.28.

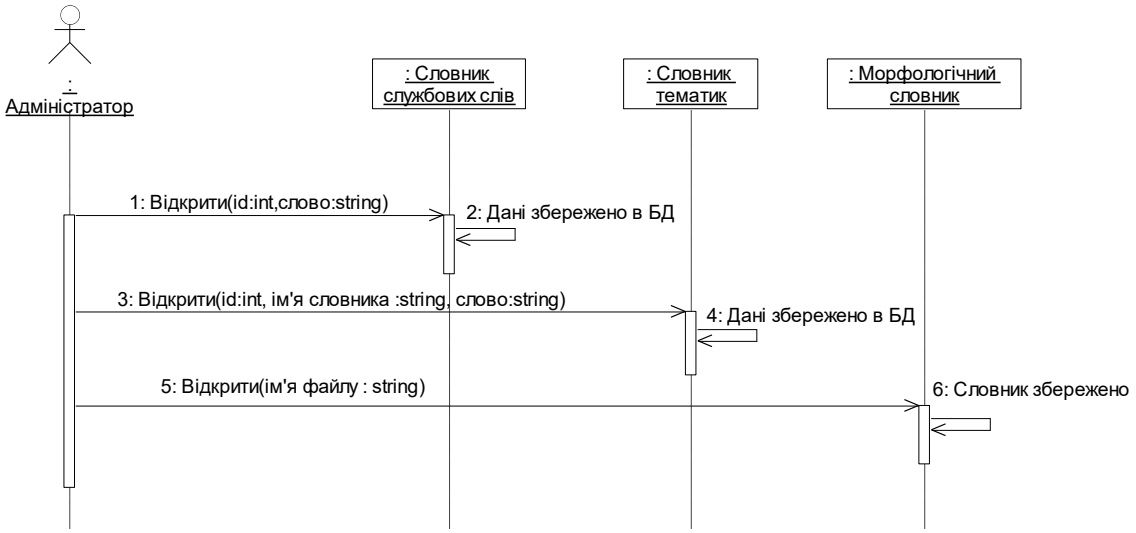


Рисунок 2.28 – Діаграма послідовності попередньої настройки системи

З діаграми видно, що перед тим як почати процес індексування текстів, необхідно заповнити БД словниками трьох типів:

1) словник службових слів;

2) словник тематик;

3) морфологічний словник.

Тепер система готова для обробки текстових документів та створення індивідуальних індексів.

На рисунку 2.29 зображена діаграма послідовності процесу індексування текстів.

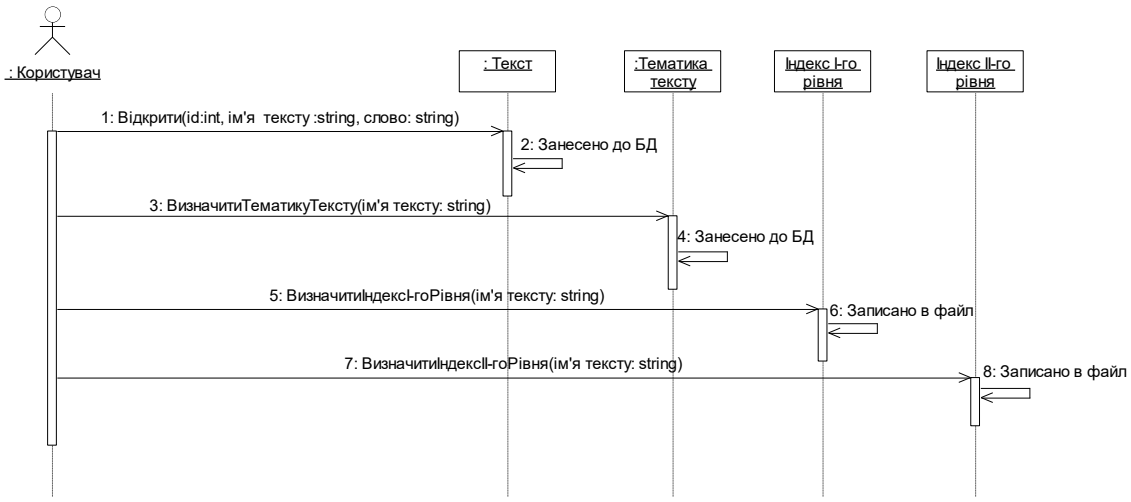


Рисунок 2.29 – Діаграма послідовності процесу індексування текстів системою

З діаграми видно які дії необхідно виконати користувачу, а саме: відкрити вибраний текст для, визначити його тематику, та створити індекси І-го та ІІ-го рівнів; і в якій послідовності.

Для тестування якості побудованих індексів, розроблено алгоритм пошуку текстів по запиту. Діаграма послідовності виконання даної операції зображена на рисунку 2.30.

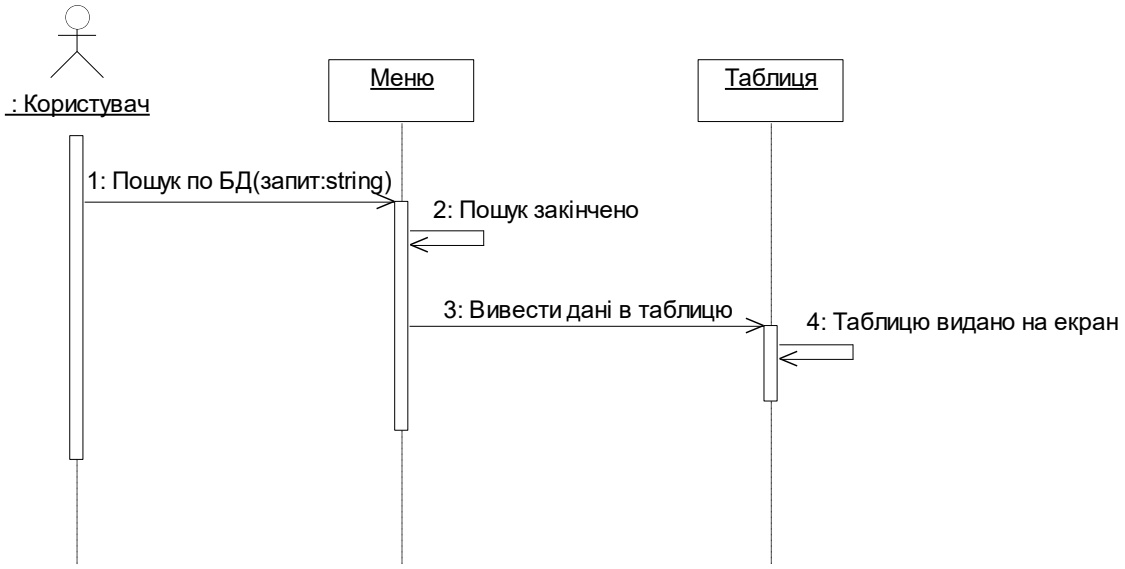


Рисунок 2.30 – Діаграма послідовності процесу пошуку по створеним індексам

Користувачу пропонується ввести строку запиту, після чого система її опрацьовує і видає результат – тексти в яких зустрічаються пошукові слова, чи слово.

Розроблена система є відкритою для користувача. Надається можливість самостійно, без втручання спеціалістів, привести систему в початковий стан. Тобто, очистити БД від словників, індексів та текстів. Діаграма послідовності даного процесу зображена на рисунку 2.31.

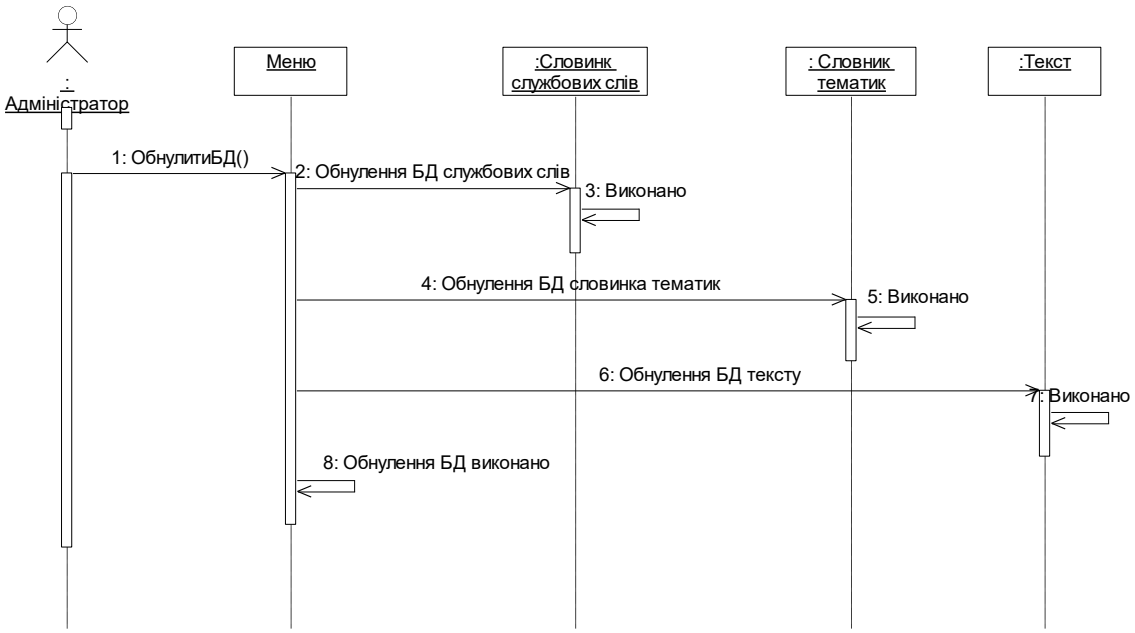


Рисунок 2.31 – Діаграма послідовності процесу очищення БД

2.6 Діаграма специфікацій системи індексування текстів

В даному підрозділі представлена діаграма специфікацій системи. Вона розділена на дві частини :

– основна форма, та класи, що напряму зв’язані з нею (Рисунок 2.32);

– класи, зв’язані з основною формою через інтерфейс меню (Рисунок 2.33).

Також представлені основні методи класів та таблиці БД.

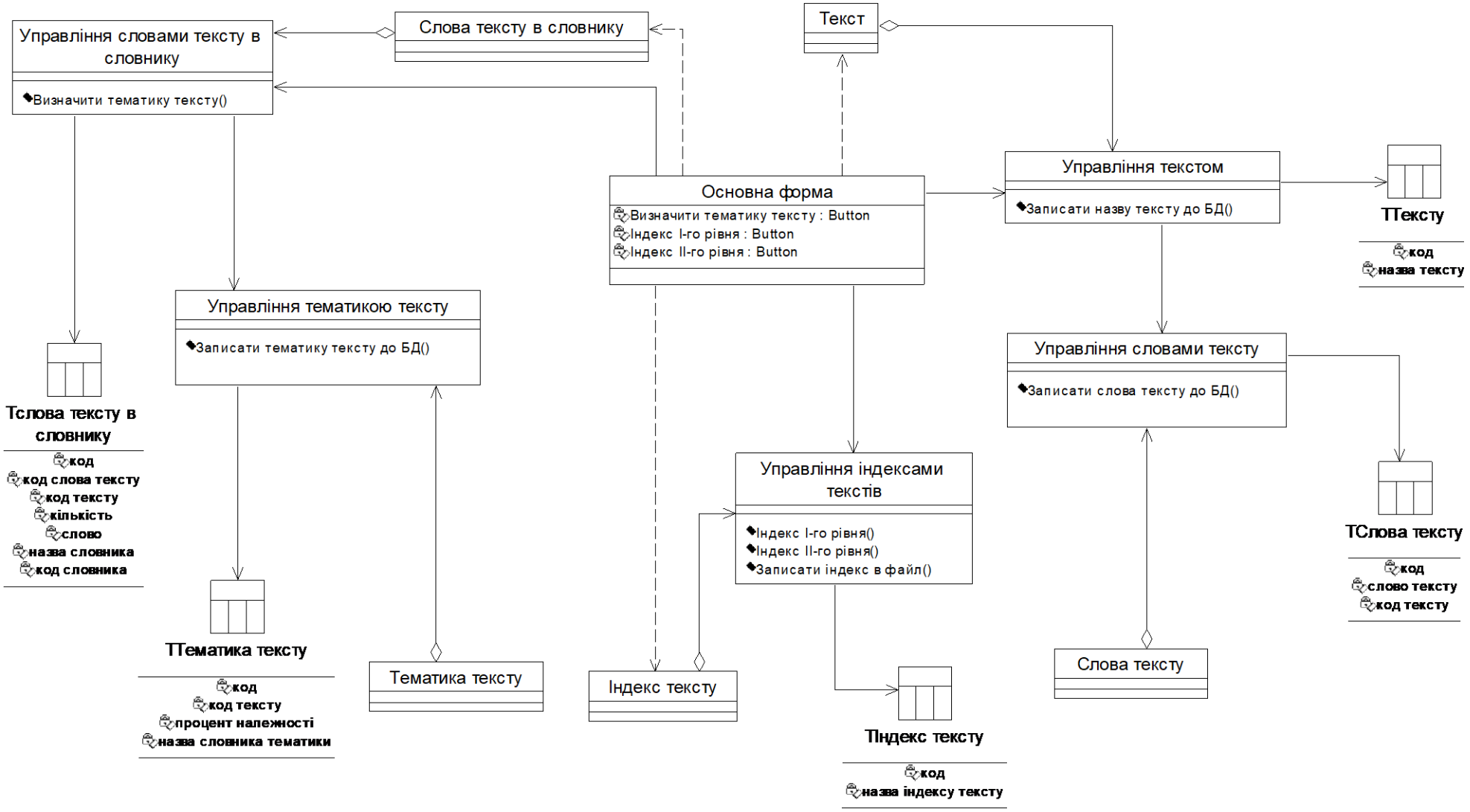


Рисунок 2.32 – Діаграма специфікацій системи індексування текстів «Основна форма»

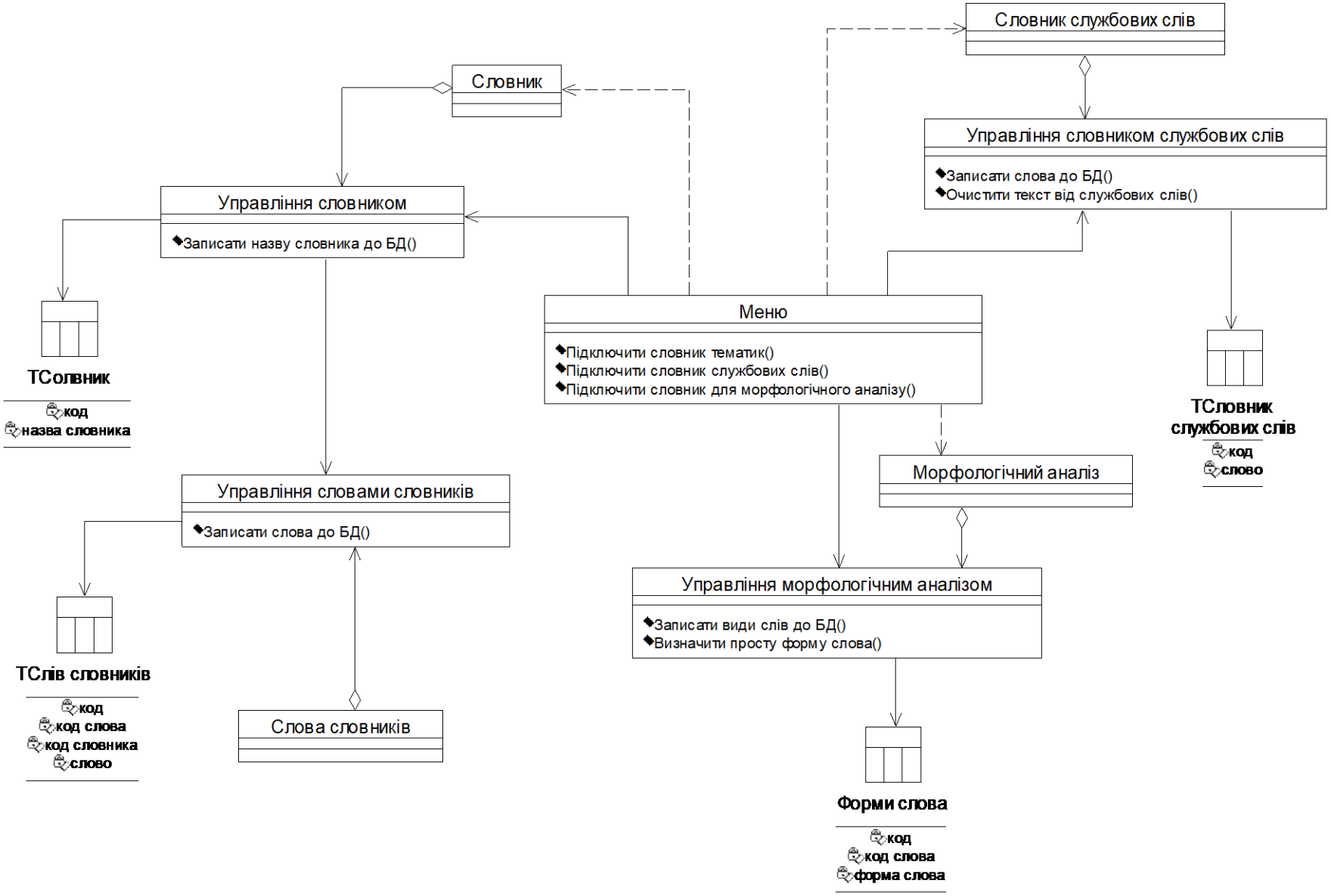


Рисунок 2.33 – Діаграма специфікацій системи індексування текстів – «Меню» основної форми

2.7 Структура БД системи індексування текстів

Для рішення поставленої задачі необхідно було розробити БД системи. На рисунку 2.34 зображені таблиці БД, побудовані в пакеті Enterprise Architect, та вказаний зв'язок між ними.

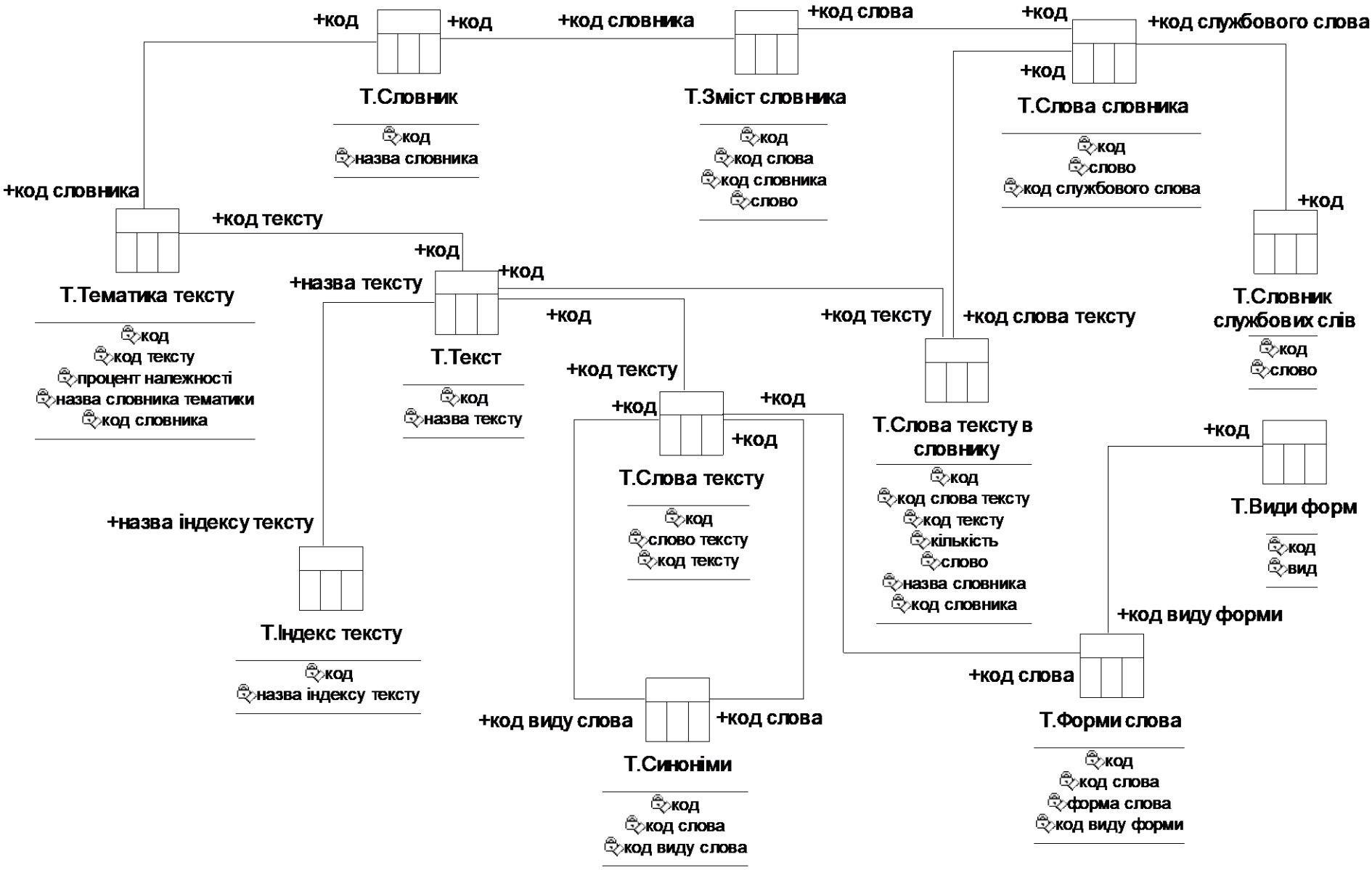


Рисунок 2.34 – БД системи індексування текстів

2.7.1 Таблиця «Т.Словник»

Для того, щоб система могла визначати предметну область текстів, необхідно загрузити в БД словники тематик. Таблиця «tDictionary» являє собою множину назв словників предметних областей (Рисунок 2.35).

Рисунок 2.35 – Таблиця словників тематик – «Т.Словник»

Як видно з рисунка 2.35, таблиця має два поля:

- код: ідентифікатор словника;
- назва словника: назва словника предметної області.

2.7.2 Таблиця «Т.Словник службових слів»

Перед тим як почати аналізувати текст, його необхідно профільтрувати на службові слова, котрі не несуть в собі важливої інформації. Таблиця «Т.Словник службових слів» містить в собі 56 службових слів (Рисунок 2.36).

Рисунок 2.36 – Таблиця службових слів - «Т.Словник службових слів»

Таблиця містить в собі два поля:

– код: ідентифікатор слова;

– слово: службове слово.

2.7.3 Таблиця «Т.Слова словника»

В даній таблиці знаходиться множина всіх слів словників предметних областей. Вона має два поля (Рисунок 2.37):

– код: ідентифікатор слова;

– слово: слово словника;

– код службового слова: ідентифікатор службового слова з таблиці «Т.Словник службових слів».

Рисунок 2.37 – Таблиця слів словників предметних областей – «Т.Слова словника»

2.7.4 Таблиця «Т.Зміст словника»

Дана таблиця містить в собі наступні поля (Рисунок 2.38):

– код: ідентифікатор;

– код слова: ідентифікатор слова з таблиці «Т.Слова словника»;

– код словника: ідентифікатор словника тематик з таблиці «Т.Словник»;

– слово: слово.

Рисунок 2.38 – Таблиця слів словників – «Т.Зміст словника»

Завдяки даній таблиці, можна дізнатися яке слово в якому словнику присутнє. Вона використовується при аналізі кожного слова тексту. Беремо слово, і перевіряємо чи присутнє воно в даній таблиці. Якщо так, то в яких словника предметних областей. Потім ці дані використовуються для визначення предметної області тексту.

2.7.5 Таблиця «Т.Текст»

Таблиця «Т.Текст» містить в собі множину назв текстів, котрі оброблені системою. Включає в себе два поля:

- код: ідентифікатор тексту;
- назва тексту: назва тексту.

Таблиця зображена на рисунку 2.39.

Рисунок 2.39 – Таблиця текстів, оброблених системою – «Т.Текст»

2.7.6 Таблиця «Т.Слова тексту»

В дану таблицю записуються всі слова поточного тексту, після того, як він відфільтрується на службові терміни, та пройде морфологічний аналіз. Записані слова в тій послідовності, в якій вони знаходяться в текстів. Таблиця має два поля (Рисунок 2.40):

– код: ідентифікатор;

– слово тексту: слово тексту;

– код тексту: ідентифікатор тексту в якому дане слово знаходиться. Взятий з таблиці «Т.Текст».

Рисунок 2.40 – Таблиця слів поточного тексту – «Т.Слова тексту»

2.7.7 Таблиця «Т.Слова тексту в словнику»

В дану таблицю записуються слова тексту і вказується в якому словнику предметної області це слово присутнє (Рисунок 2.41).

Рисунок 2.41 – Таблиця слів тексту з вказаним словником тематики, де воно присутнє – «Т.Слова тексту в словнику»

В таблиці присутні наступні поля:

– код: ідентифікатор;

– код слова тексту: ідентифікатор слова з таблиці «Т.Слова тексту»;

– код тексту: ідентифікатор тексту, в якого взяте дане слово. Ідентифікатор тексту береться з таблиці «Т.Текст»;

– кількість: кількість повторень слова в тексті з ідентифікатором «код тексту»;

– слово: слово тексту з ідентифікатором «код тексту»;

– назва словника: назва словника предметної області, в якому дане слово присутнє;

– код словника: ідентифікатор словника, в якому дане слово присутнє.

2.7.8 Таблиця «Т.Тематика тексту»

В таблицю «Т.Тематика тексту» записується процентне співвідношення належності тексту до тієї чи іншої предметної області (Рисунок 2.42).

Рисунок 2.42 – Таблиця процентного співвідношення належності тексту до тієї чи іншої предметної області

В таблиці присутні наступні поля:

– код: ідентифікатор;

– код тексту: ідентифікатор тексту, з таблиці «Т.Текст»;

– процент належності: число належності тексту до предметної області в процентах;

– назва словника тематики: назва словника предметної області, до якої належить той чи інший текст.

– код словника: код словника предметної області, до якої належить той чи інший текст.

2.7.9 Таблиця «Т.Індекс тексту»

Дана таблиця містить в собі множину всіх індексів побудованих системою (Рисунок 2.43). З рисунку видно, що в таблиці присутні такі поля, як:

– код: ідентифікатор;

– назва індексу тексту: назва файлу проіндексованого тексту.

Назва файлу індексу відповідає назві тексту, для якого він створювався.

Рисунок 2.43 – Таблиця множини індексів побудованих системою

2.7.10 Таблиця «Т.Синоніми»

Таблиця призначена для опису синонімів конкретного слова. Містить такі поля (Рисунок 2.44):

– код: ідентифікатор;

– код слова: ідентифікатор слова тексту з таблиці «Т.Слова тексту»;

– код виду слова: ідентифікатор слова тексту з таблиці «Т.Слова тексту».

Рисунок 2.44 – Таблиця «Т.Синоніми»

2.7.11 Таблиця «Т.Форми слова»

Дана таблиця разом з таблицею «Т.Види форм» виконують призначені для морфологічного аналізу слів тексту (Рисунок 2.45). Як видно з рисунку, таблиця містить наступні поля:

– код: ідентифікатор;

– код слова: ідентифікатор слова з таблиці «Т.Слова тексту»;

– форма слова: словоформа слова;

– код виду форми: ідентифікатор словоформи.

Рисунок 2.45 – Таблиця «Т.Форма слова»

2.7.12 Таблиця «Т.Види форм»

Таблиця призначена для опису різних форм слова: множинних, одиничних і т д. Таблиця містить такі поля (Рисунок 2.46):

– код: ідентифікатор;
– вид: вид форми слова.

Рисунок 2.46 – Таблиця «Т.Види форм»

3 ОПИС СИСТЕМИ ІНДЕКСУВАННЯ ТЕКСТІВ

В даному розділі описується інтерфейс користувача, та база даних Access програми. Текст програми знаходиться в додатку А.

3.1 Інтерфейс користувача системи індексування текстів

Інтерфейс програми поділений на декілька пунктів меню. В даному розділі описується кожен з них.

На рисунку 3.1 зображений інтерфейс програмного продукту.

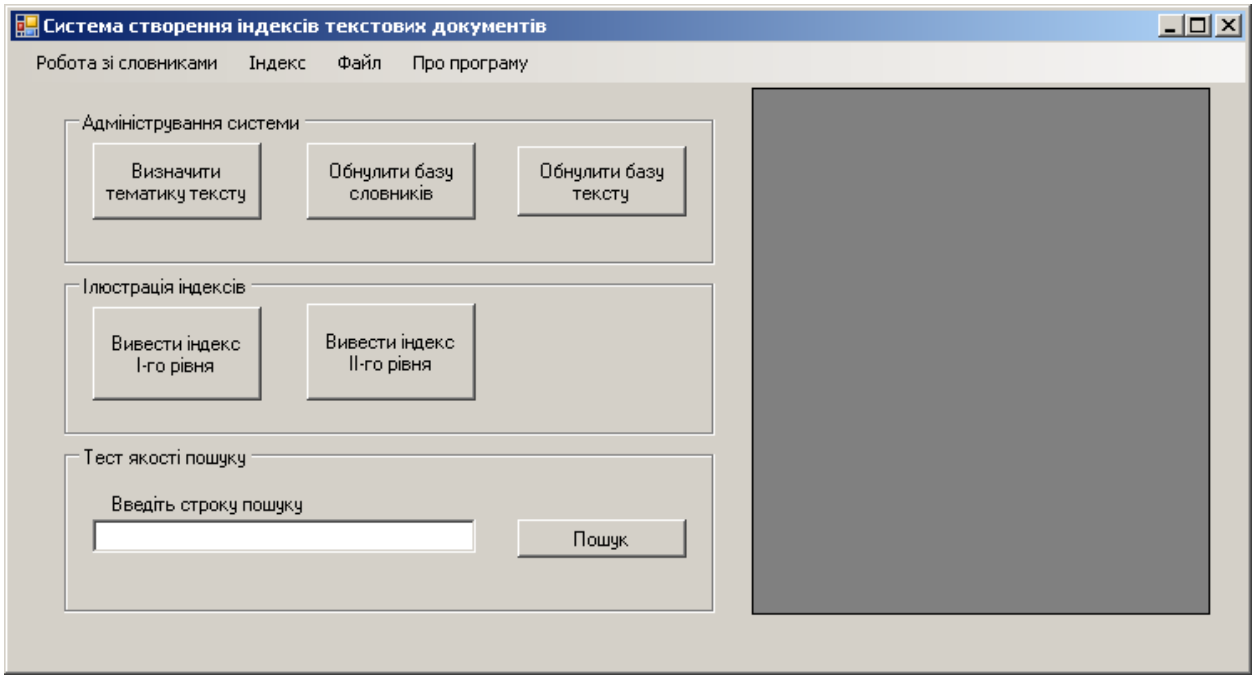


Рисунок 3.1 – Інтерфейс програми

3.2 Меню «Робота зі словниками»

«Робота зі словниками» - дане меню призначене для підключення в систему необхідних словників: 1) словник службових слів; 2) словники тематик; 3) словник для морфологічного аналізу. На рисунку 3.2 зображено підпункти меню роботи зі словниками.

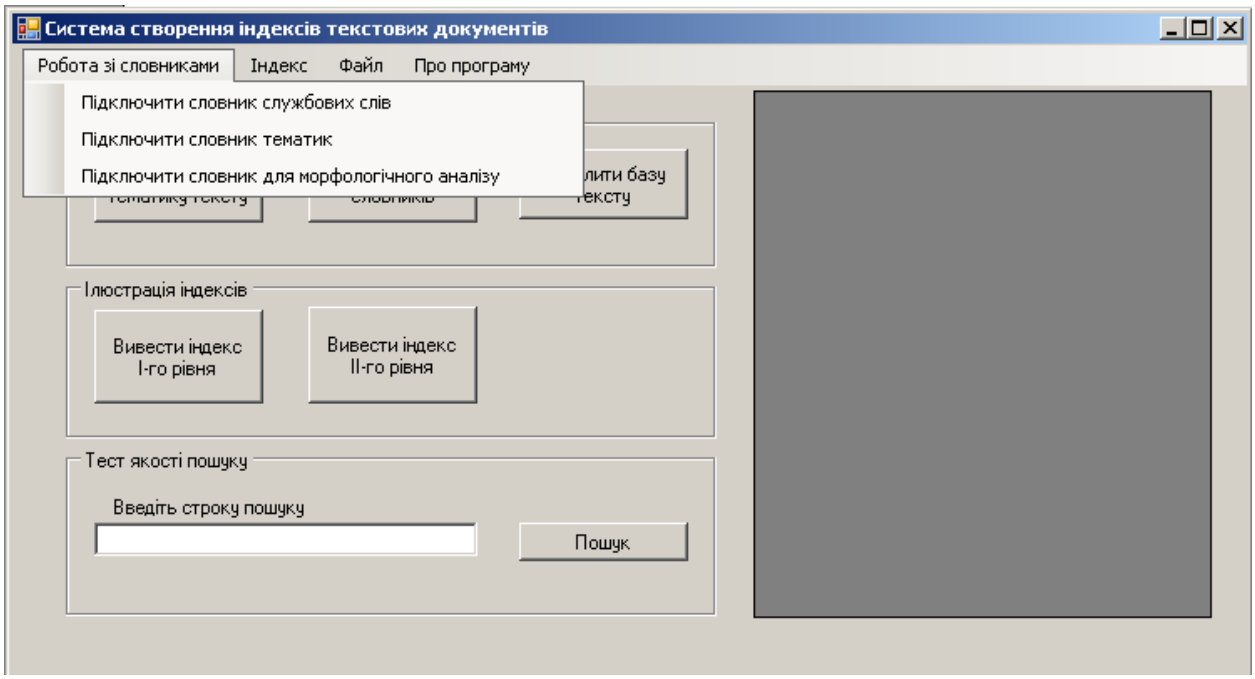


Рисунок 3.2 – Меню роботи зі словниками

В даному меню є три пункти:

– Підключити словник службових слів;
– Підключити словник тематик;
– Підключити словник для морфологічного аналізу.

3.2.1 «Підключити словник службових слів»

При натисканні даного підменю, користувачу пропонується відкрити текстовий файл в форматі .txt, після чого слова файлу записуються до БД. На рисунку 3.3 представлена таблиця службових слів – «tSlugWord». В даній версії програми, їх використовується 56.

tSlugWord : таблица

ID	Word
7	в
8	на
9	с
10	за
11	к
12	по
13	из
14	у
15	от
16	для
17	во
18	без
19	до
20	о
21	через
22	со
23	при
24	про
25	об
26	ко
27	над
28	из-за
29	из-под
30	под
31	и
32	что
33	но
34	а
35	да
36	хотя
37	когда
38	чтобы
39	если
40	тоже
41	или

Запись: 1 из 672

Рисунок 3.3 – Таблиця службових слів «tSlugWord» в БД

3.2.2 «Підключити словник тематик»

Словників тематик використовується декілька, в залежності від потреб користувача. Після вибору словника в форматі txt, та його підключення в систему, дані записуються в декілька таблиць:

– таблиця словників – «tDictionary» (Рисунок 3.4);

– таблиця слів словників - «tWordDic» (Рисунок 3.5);

– таблиця слів словників відповідних тематик – «tDicContent» (Рисунок 3.6).

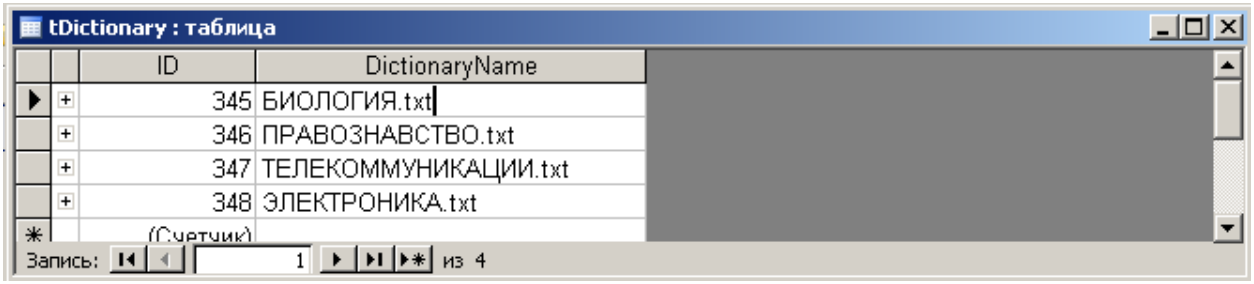


Рисунок 3.4 – Таблиця словників «tDictionary» в БД

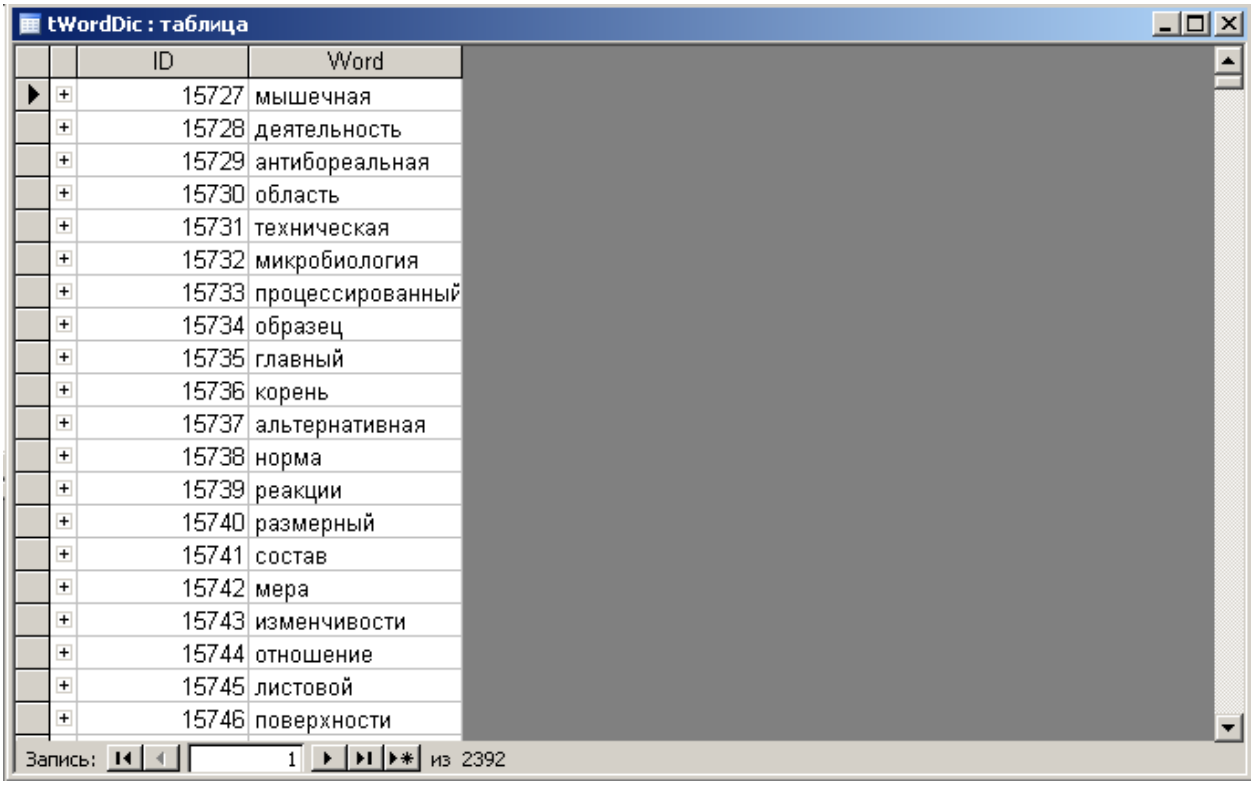


Рисунок 3.5 – Таблиця слів словників «tWordDic» в БД

tDicContent : таблица

ID	IDWord	IDDictionary	Word
11900	15727	345	мышечная
11901	15728	345	деятельность
11902	15729	345	антибореальная
11903	15730	345	область
11904	15731	345	техническая
11905	15732	345	микробиология
11906	15733	345	процессированный
11907	15734	345	образец
11908	15735	345	главный
11909	15736	345	корень
11910	15737	345	альтернативная
11911	15738	345	норма
11912	15739	345	реакции
11913	15740	345	размерный
11914	15741	345	состав
11915	15742	345	мера
11916	15743	345	изменчивости
11917	15744	345	отношение
11918	15745	345	листовой

Запись: 1 из 2909

Рисунок 3.6 – Таблиця слів словників відповідних тематик «tDicContent» в БД

3.2.3 «Підключити словник морфологічного аналізу»

Словник словоформ використовується в вигляді текстового файлу. Містить в собі 1 200 000 словоформ. Приклад словника зображено на рисунку 3.7.

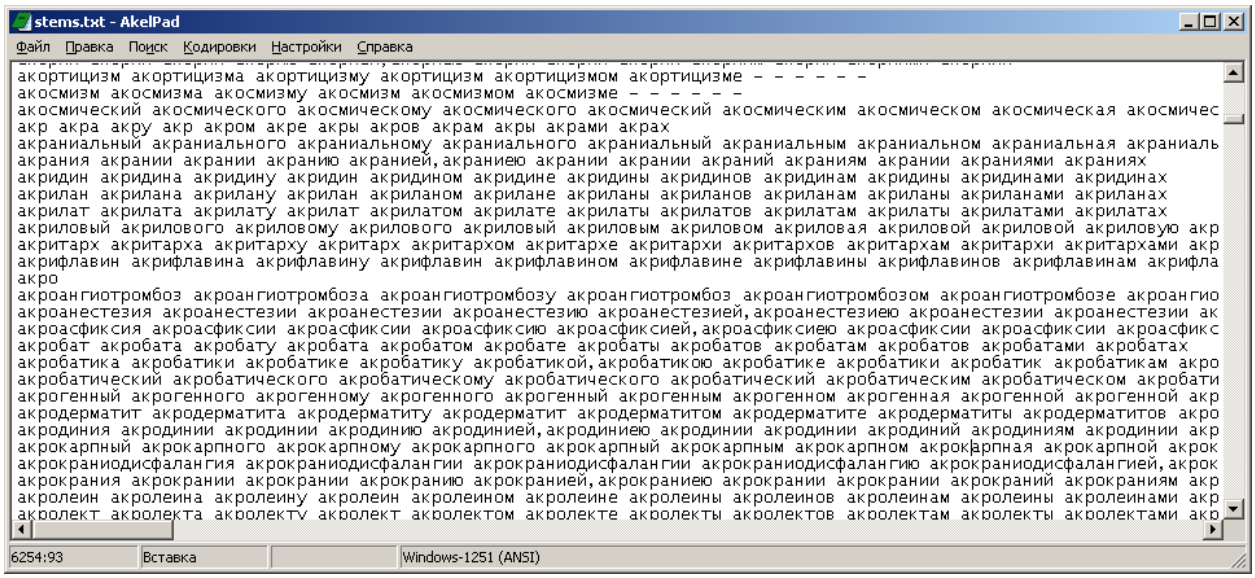


Рисунок 3.7 – Словник словоформ

3.3 Меню «Файл»

Дане меню містить в собі два підпункти (Рисунок 3.8):

– «Відкрити текст»: для роботи з текстовими документами, які необхідно про індексувати;

– «Вихід»: для виходу з системи.

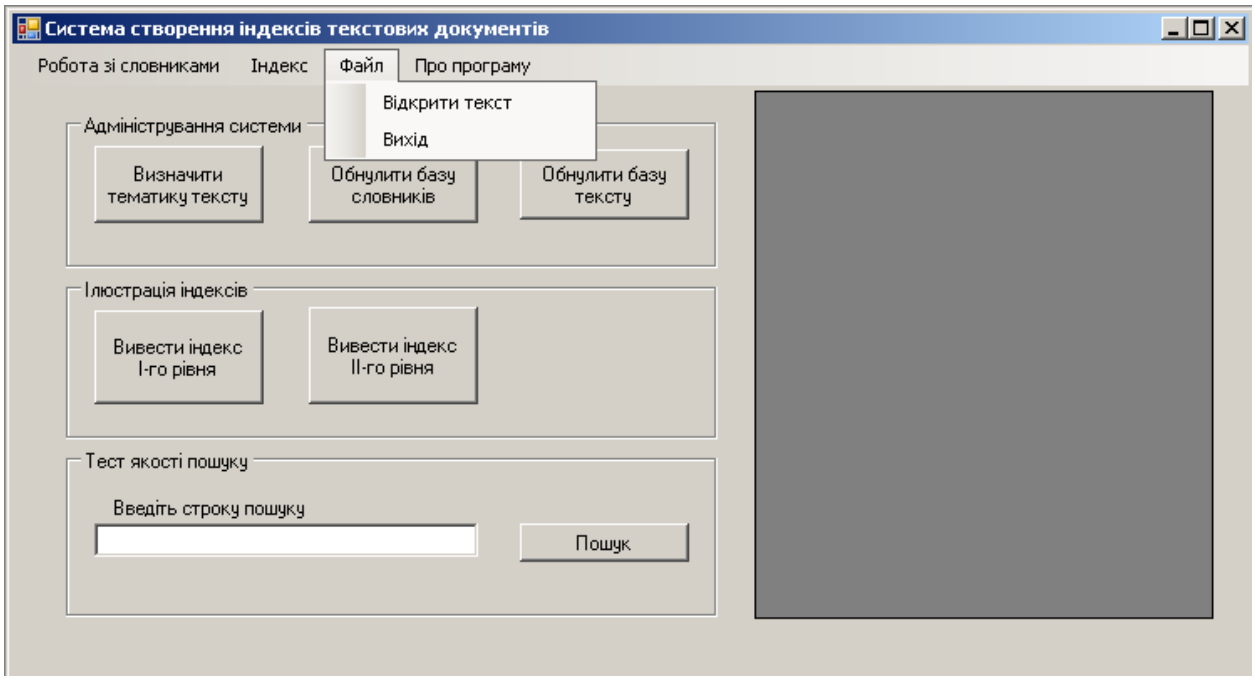


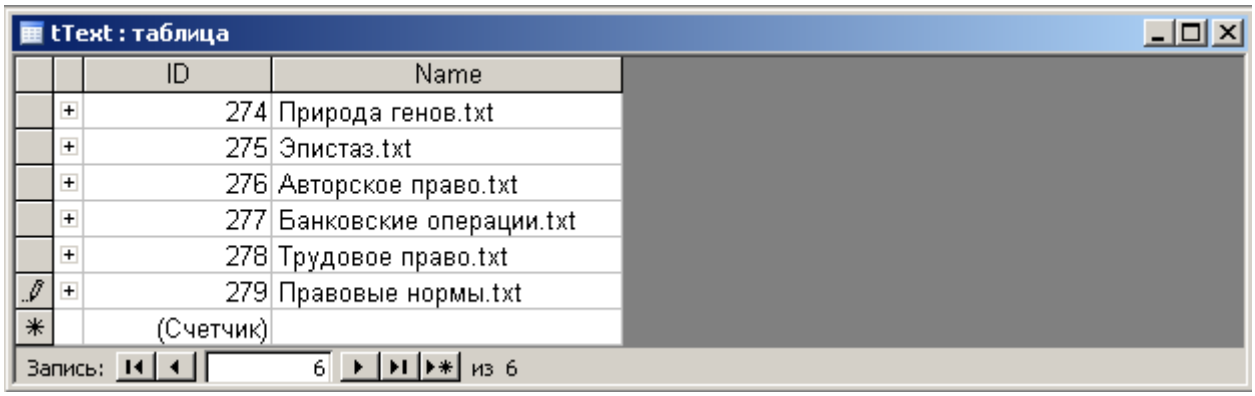
Рисунок 3.8 – Меню «Файл»

3.3.1 «Відкрити текст»

Після того, як всі необхідні словники загружено до БД, можна відкривати для створення індексу текстові документи. При виборі даного підменю відкривається діалогове вікно, де пропонується вибрати текстовий документ для обробки. Вибравши текст і натиснувши кнопку «ОК» відбувається видалення з вибраного документу службових слів, та морфологічний аналіз кожного з них. Потім дані записуються в декілька таблиць, а саме:

– таблиця текстів – «tText» (Рисунок 3.9);

– таблиця слів тексту – «tWordText» (Рисунок 3.10).

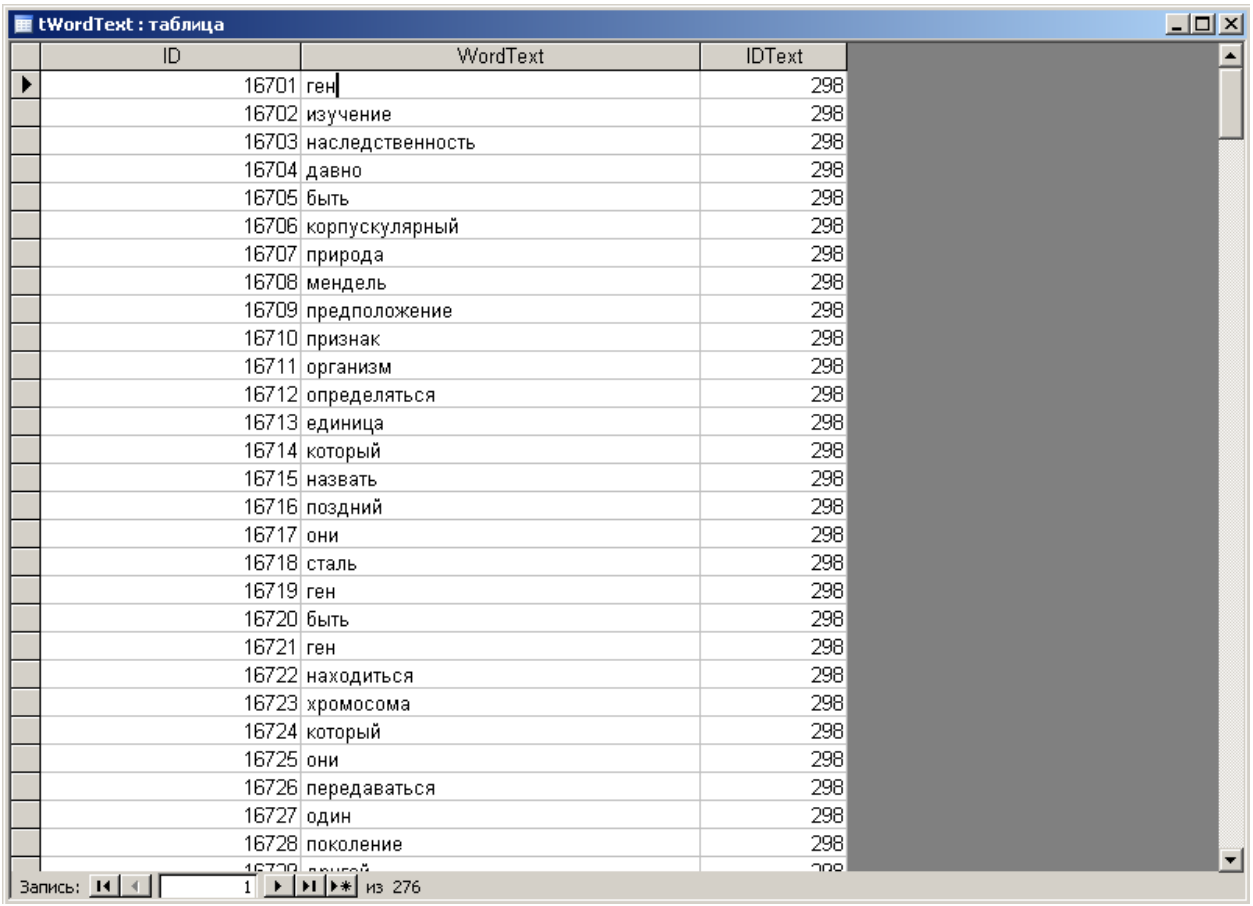


tText : таблица

	ID	Name
+	274	Природа генов.txt
+	275	Эпистаз.txt
+	276	Авторское право.txt
+	277	Банковские операции.txt
+	278	Трудовое право.txt
+	279	Правовые нормы.txt
*	(Счетчик)	

Запись: 6 из 6

Рисунок 3.9 – Таблиця текстів «tText» в БД



tWordText : таблица

ID	WordText	IDText
16701	ген	298
16702	изучение	298
16703	наследственность	298
16704	давно	298
16705	быть	298
16706	корпускулярный	298
16707	природа	298
16708	мендель	298
16709	предположение	298
16710	признак	298
16711	организм	298
16712	определяться	298
16713	единица	298
16714	который	298
16715	назвать	298
16716	поздний	298
16717	они	298
16718	сталь	298
16719	ген	298
16720	быть	298
16721	ген	298
16722	находиться	298
16723	хромосома	298
16724	который	298
16725	они	298
16726	передаваться	298
16727	один	298
16728	поколение	298

Запись: 1 из 276

Рисунок 3.10 – Таблиця слів тексту з виділеними службовими словами, та після морфологічного аналізу - «tWordText» в БД

3.4 Кнопка «Визначити тематику тексту»

Після того, як текст оброблений і записаний послівно до БД, необхідно визначити його тематику.

Для цього необхідно натиснути на кнопку «Визначити тематику тексту». Відбуваються наступні процеси:

– визначаються всі словники тематик, до яких належать слова тексту. Ці дані записуються в таблицю «tWordTextDicContent» (Рисунок 3.11);

– в таблиці «tTextSubject» знаходиться процентне співвідношення належності тексту до тієї чи іншої предметної області (Рисунок 3.12).

tWordTextDicContent : таблица

ID	IDWordText	IDText	Kil	Word	DicName	IDDictionary
8547	16701	298	21	ген	БИОЛОГИЯ.txt	345
8548	16723	298	14	хромосома	БИОЛОГИЯ.txt	345
8549	16727	298	8	один	БИОЛОГИЯ.txt	345
8550	16727	298	8	один	ПРАВОЗНАВСТВО.txt	346
8551	16727	298	8	один	ЭЛЕКТРОНИКА.txt	348
8552	16705	298	8	быть	ПРАВОЗНАВСТВО.txt	346
8553	16759	298	7	этот	ПРАВОЗНАВСТВО.txt	346
8554	16920	298	7	тело	БИОЛОГИЯ.txt	345
8555	16920	298	7	тело	ПРАВОЗНАВСТВО.txt	346
8556	16920	298	7	тело	ЭЛЕКТРОНИКА.txt	348
8557	16710	298	6	признак	БИОЛОГИЯ.txt	345
8558	16710	298	6	признак	ПРАВОЗНАВСТВО.txt	346
8559	16710	298	6	признак	ЭЛЕКТРОНИКА.txt	348
8560	16713	298	5	единица	БИОЛОГИЯ.txt	345
8561	16713	298	5	единица	ПРАВОЗНАВСТВО.txt	346
8562	16713	298	5	единица	ТЕЛЕКОММУНИКАЦИИ.txt	347
8563	16713	298	5	единица	ЭЛЕКТРОНИКА.txt	348
8564	16761	298	4	участок	БИОЛОГИЯ.txt	345
8565	16761	298	4	участок	ПРАВОЗНАВСТВО.txt	346
8566	16761	298	4	участок	ТЕЛЕКОММУНИКАЦИИ.txt	347
8567	16761	298	4	участок	ЭЛЕКТРОНИКА.txt	348
8568	16717	298	4	они	ПРАВОЗНАВСТВО.txt	346
8569	16750	298	4	основание	ПРАВОЗНАВСТВО.txt	346
8570	16750	298	4	основание	ТЕЛЕКОММУНИКАЦИИ.txt	347
8571	16750	298	4	основание	ЭЛЕКТРОНИКА.txt	348
8572	16865	298	3	такой	ПРАВОЗНАВСТВО.txt	346
8573	16842	298	3	сцепление	БИОЛОГИЯ.txt	345

Запись: 1 из 189

Рисунок 3.11 – Таблиця належності слів тексту до словників тематик

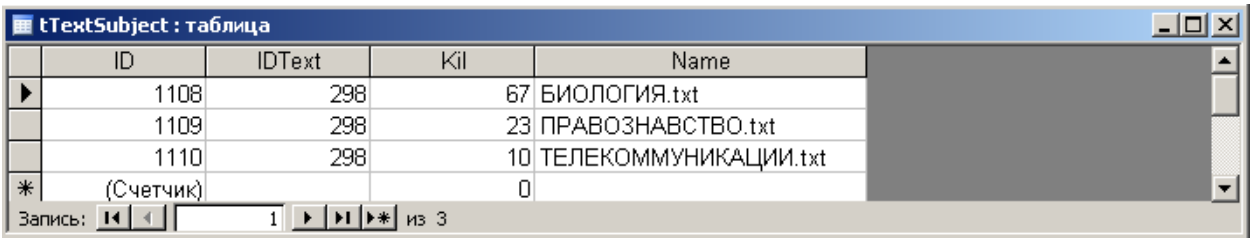


Рисунок 3.12 – Таблиця тематики тексту «tTextSubject»

3.5 Меню «Індекс»

Для того, щоб побудувати індекс для поточного тексту, необхідно звернутися до меню «Індекс», його підпункти зображені на рисунку 3.13.

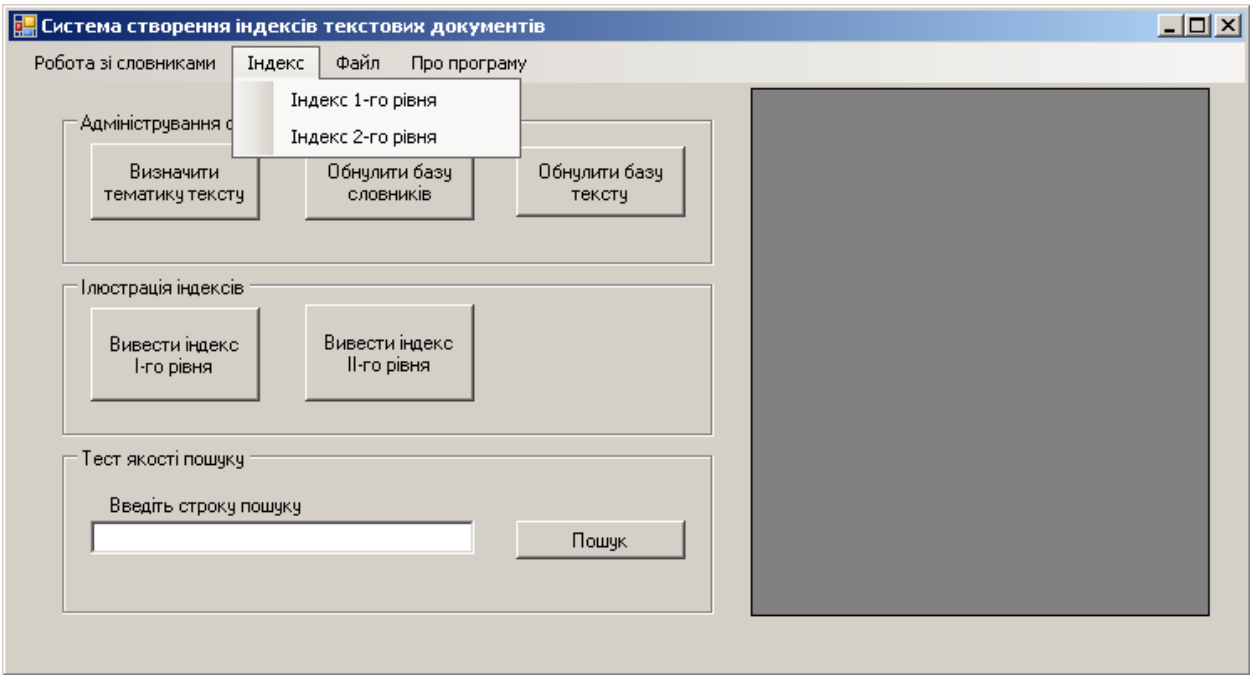


Рисунок 3.13 – Підпункти меню «Індекс»

При виборі конкретного підпункту, відбувається наступне: в текстовий файл з назвою поточного тексту зберігається відповідний індекс, також побудований індекс виводиться в вигляді таблиці в вікно зображене справа.

Для того, щоб проглянути індекс I-го рівня необхідно натиснути на кнопку «Вивести індекс I-го рівня» (Рисунок 3.14).

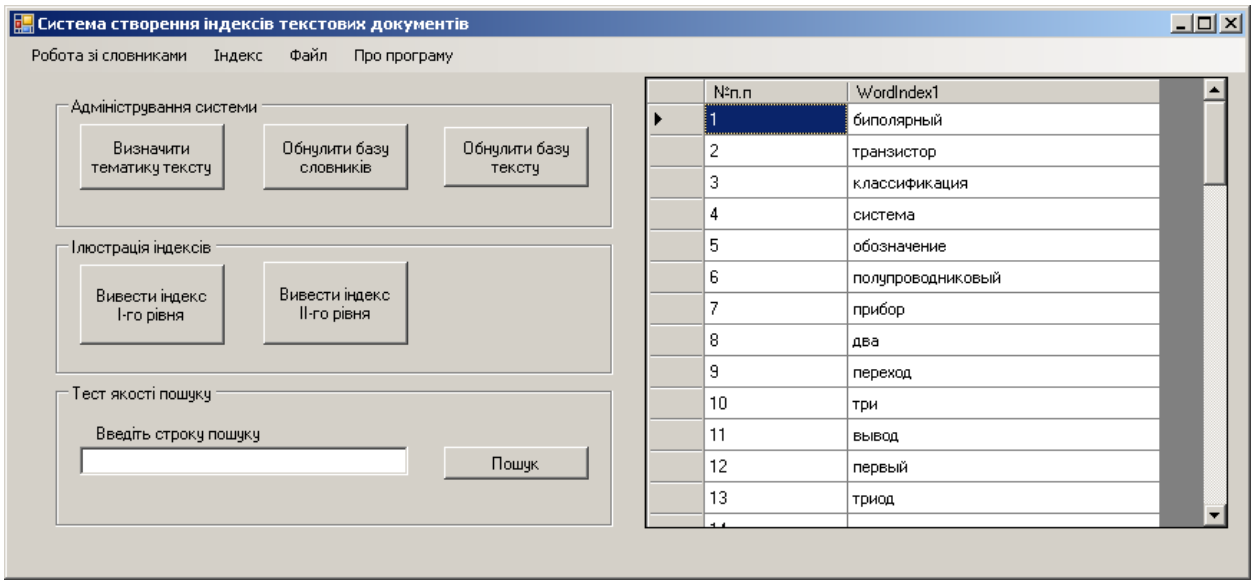


Рисунок 3.14 – Вивід в таблицю індексу I-го рівня

Для того, щоб проглянути індекс II-го рівня необхідно натиснути на кнопку «Вивести індекс II-го рівня» (Рисунок 3.15).

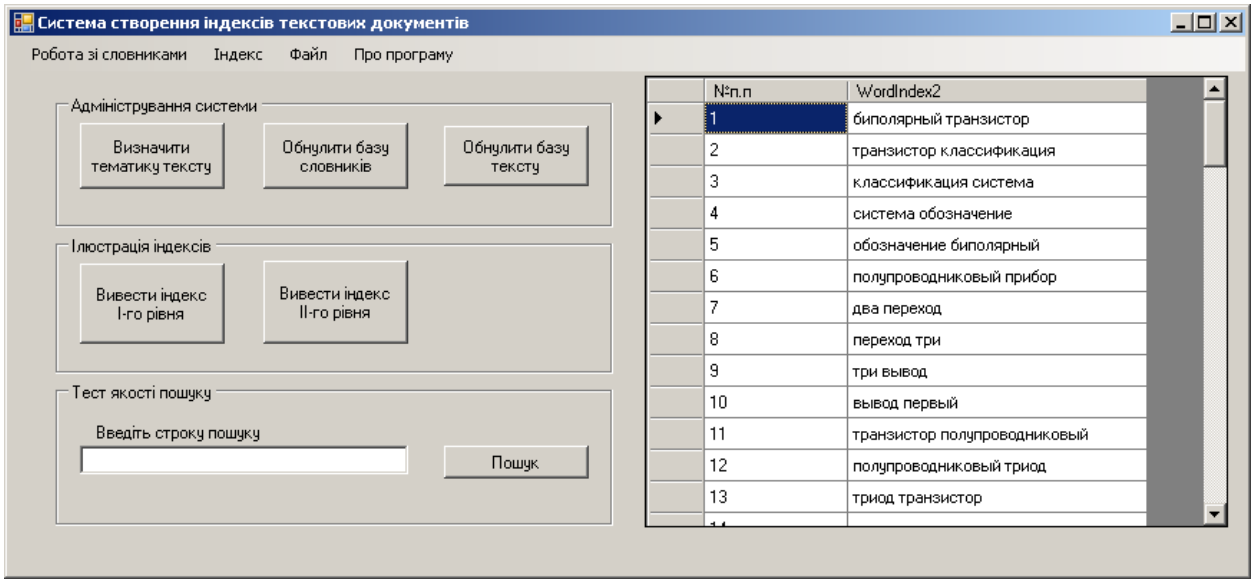


Рисунок 3.15 – Вивід в таблицю індексу II-го рівня

3.6 Кнопка «Пошук»

Для перевірки якості індексування системи, була розроблена підсистема для виконання пошуку запиту користувача по індексах.

Якщо користувач вводить одне слово в строку пошуку, то відбувається пошук по індексах першого рівня, і результат виводиться вікно програми (Рисунок 3.16).

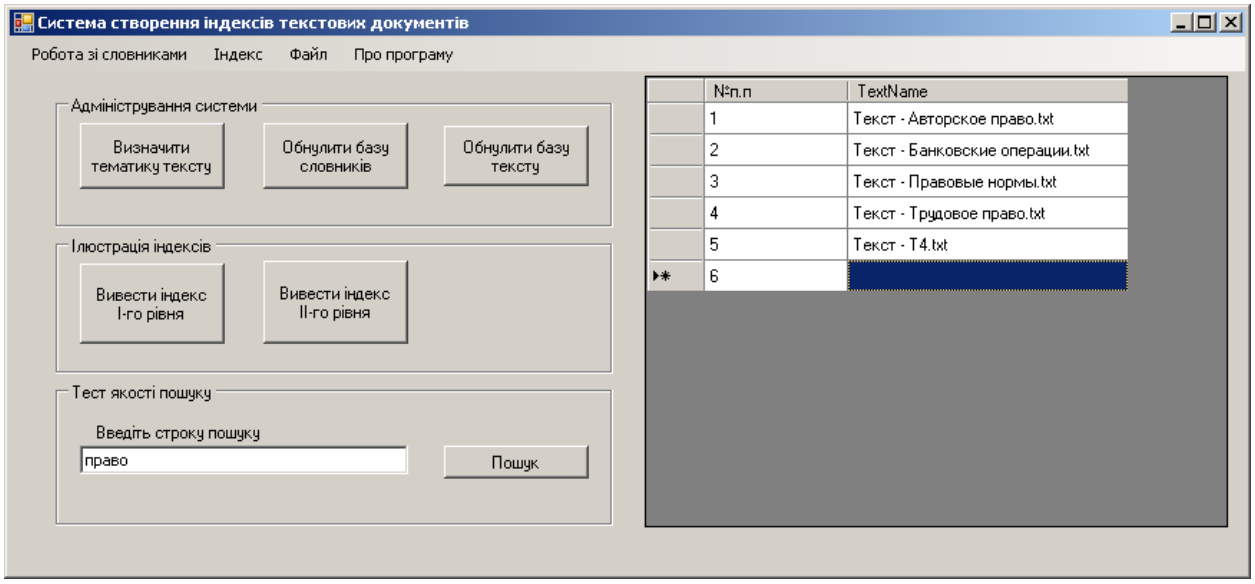


Рисунок 3.16 – Результат пошуку запиту користувача по індексах I-го рівня

При вводі більш ніж одного слова в строку пошуку, відбувається пошук по індексах II-го рівня. Результат також виводиться в вікно програми (Рисунок 3.17).

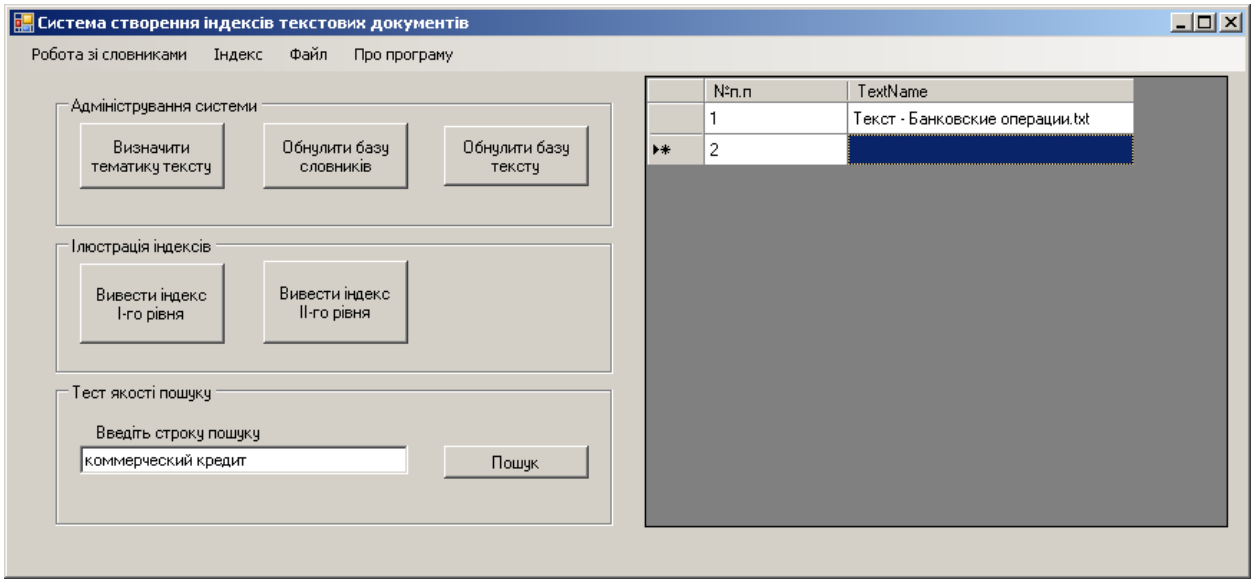


Рисунок 3.17 – Результат пошуку запиту користувача по індексах II-го рівня

3.7 БД системи

Для рішення поставленої задачі необхідно було розробити БД системи. На рисунку 3.18 зображені таблиці БД, побудованої в Access, та вказаний зв'язок між ними. Всі таблиці описані в попередньому розділі.

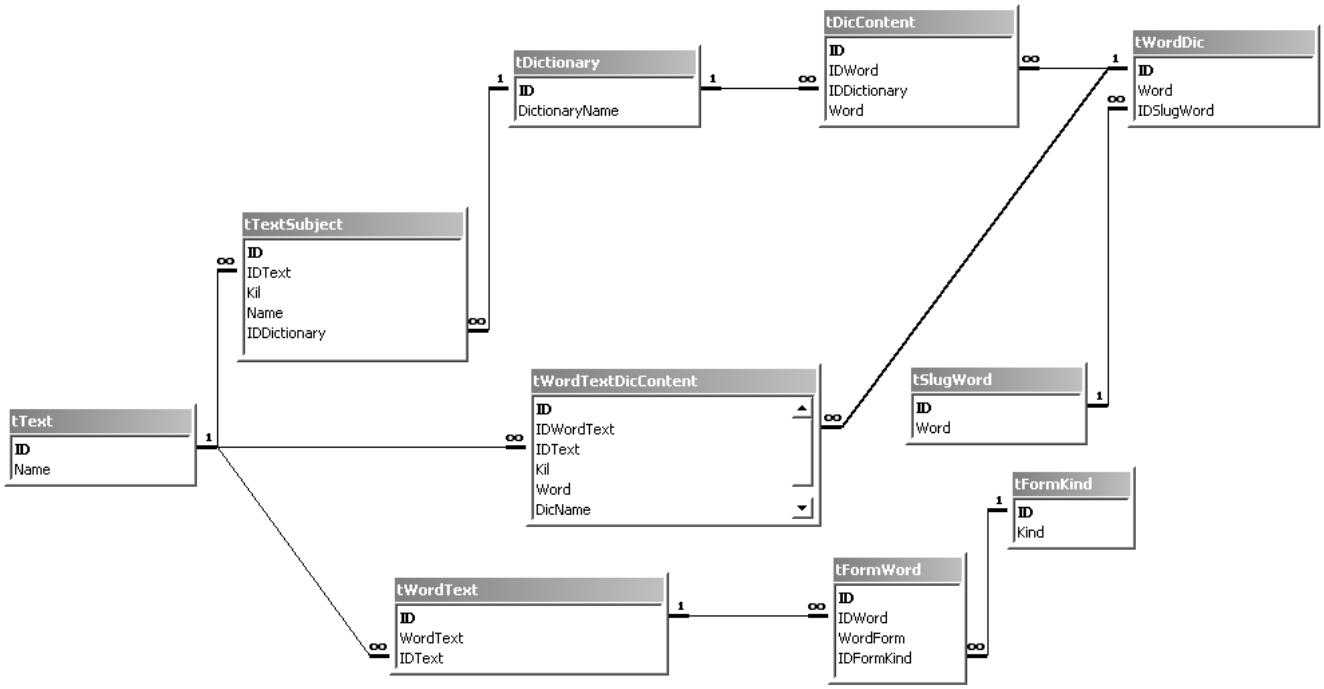


Рисунок 3.18 – БД системи індексування текстів

4 ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

Розроблений механізм індексування текстів є модифікацією двох існуючих алгоритмів: координатного та класифікаційного. В даному розділі приводяться результати роботи кожного з них.

Якщо використовувати класифікаційне індексування, тексти, в залежності від їх змісту, відносяться до відповідного класу в якому накопичуються всі тексти, що мають схожий зміст. Кожному такому тексту відповідає індекс даного класу, що і виступає його пошуковим образом. Недоліком є те, що тексти можуть відрізнятися і при цьому мати один загальний індекс.

4.1 Експеримент №1

Для прикладу, візьмемо шість текстів. Чотири з них – тексти з правознавства, і два – з біології. Всі тексти знаходяться в додатку.

Кожен з вибраних текстів з правознавства відноситься до певної підтеми даної предметної області. Серед них:

- «Авторское право»;
- «Банковские операции»;
- «Правовые нормы»;
- «Трудовое право».

Тексти з біології також описують різні біологічні процеси:

- «Природа генов»;
- «Эпистаз».

4.1.1 Класифікаційне індексування

Алгоритм роботи має наступні кроки:

Крок 1 Побудувати індекси відповідних класів. Кожен клас – це предметна область. Індекс створюється з основних слів, які найчастіше вживаються в даній предметній області.

Крок 2 Подаємо на вхід системи текст, індекс якого необхідно побудувати.

Крок 3 Визначаємо предметну область тексту, тобто клас до якого він належить.

Крок 4 В залежності від результатів виконання кроку 3, даному тексту назначається індекс відповідного класу.

При класифікаційному індексуванні тексти, в залежності від змісту відносяться до відповідного класу. Нехай, буде клас «Правознавство» у якого є свій індекс, який присвоюється до текстів віднесених до нього.

На рисунку 4.1 зображено інтерфейс програми, де користувачу пропонується ввести строку пошуку.

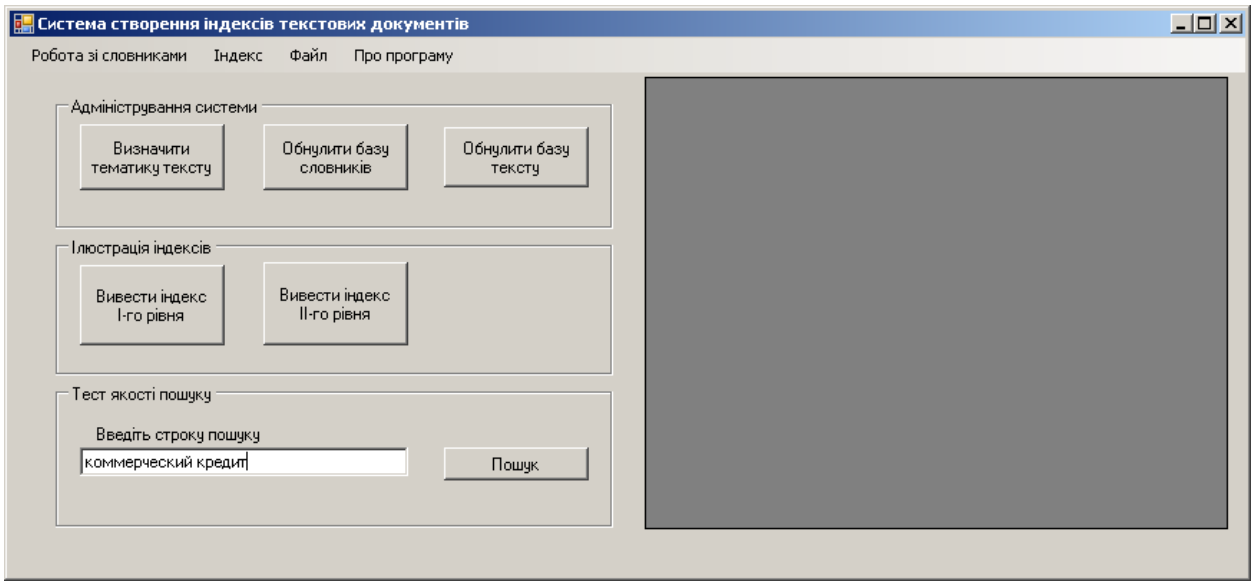


Рисунок 4.1 – Введення запиту на пошук при використанні класифікаційного алгоритму індексування

Натиснувши на кнопку «Пошук», в правому вікні програми бачимо результат пошуку, який зображено на рисунку 4.2.

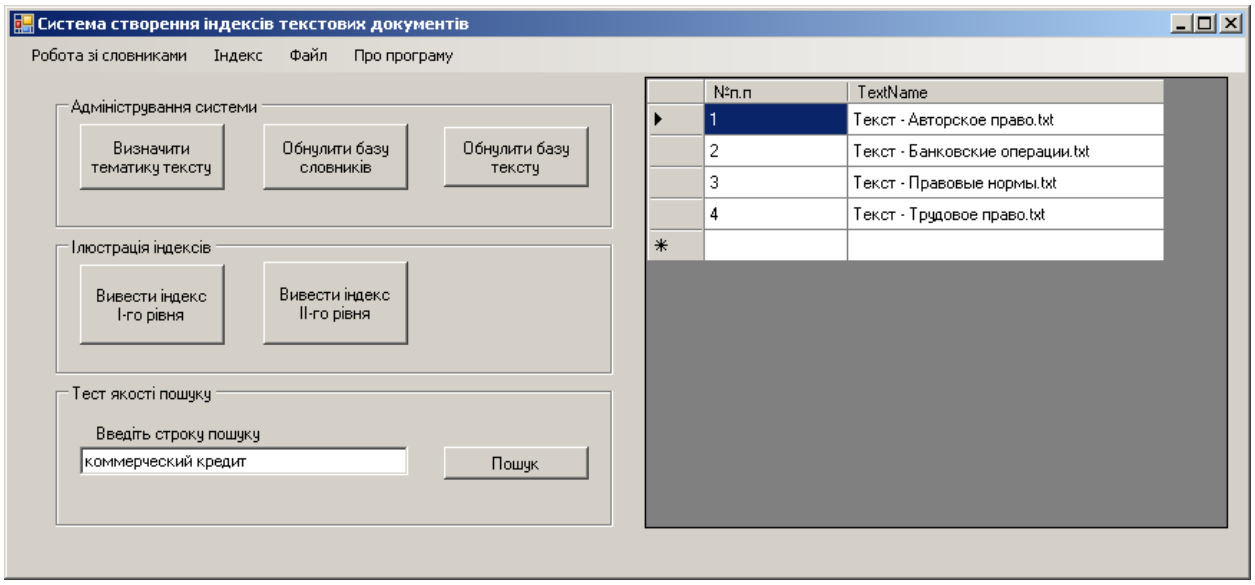


Рисунок 4.2 – Результат пошуку з строкою запиту «коммерческий кредит»

Як видно з рисунку 4.2 на строку запиту «коммерческий кредит» результатом являються всі тексти, що відносяться до класу «Правознавство».

4.1.2 Координатне індексування текстів

Алгоритм координатного індексування документів оснований на врахуванні класифікаційних характеристик присутніх в тексті термінів (слів та словосполучень), характеризуючи ту чи іншу предметну область. Для цього необхідно створення словника термінів предметної області, при чому, в цьому словнику повинні бути установлені зв’язки між термінами та проведена їх класифікація. Такий словник називається тезаурусом. В якості списка ключових слів та словосполучень для тезауруса пропонується використовувати предметний показник спеціалізованої енциклопедії. Вибір конкретної енциклопедії виконує спеціаліст з предметної області, і цей вибір залежить від цілей, які поставлені при створенні тезауруса. В якості дескрипторів (термінів, що являються іменами класів близьких по суті понять) використовуються назви статей енциклопедій, а зв’язаними з ними

по суті вважаються слова з предметного показника, що зустрічаються в відповідних статтях.

Завдяки тезаурусу, даний алгоритм має набагато чіткіший індекс ніж класифікаційне індексування. Тобто результат пошуку буде теж точнішим. Але за рахунок врахування зв'язків між словами в словосполученні, швидкість пошуку від цього повільніша ніж в класифікаційного алгоритму.

4.1.3 Механізм дворівневого індексування

Розглянемо ще раз коротко алгоритм роботи системи дворівневого індексування:

Крок 1 Підключаємо словники

Крок 1.1 Підключаємо словники тематик

Крок 1.2 Підключаємо словник службових слів

Крок 2 Відкриваємо текст, індекс якого будемо створювати.

Крок 3 Очищаємо поточний текст від службових слів.

Крок 4 Виконуємо морфологічний аналіз відфільтрованих слів поточного тексту.

Крок 5 Будуємо базис з відфільтрованих слів поточного тексту. Тобто фіксуємо яке слово тексту, скільки раз зустрічається, і в якому словнику присутнє.

Крок 6 Виходячи з даних отриманих на попередньому кроці, визначаємо вірогідність належності тексту до тієї чи іншої тематики.

Крок 7 Після визначення тематики, із словників предметних областей використовується лише той, до предметної області якого належить текст.

Крок 8 Будуємо індекс першого рівня. Ним являються слова тексту, що присутні в словнику тематики, до якої він належить.

Крок 9 Будуємо індекс другого рівня. Беремо послідовно всі

словосполучення з відфільтрованого тексту після Кроку 4, і якщо обидва слова словосполучення присутні в словнику тематики – зберігаємо його в файл. При пошуку текстів відповідно запиту, спочатку сканується індекс другого рівня, оскільки він являються більш чітким. Якщо результат буде нульовий, то сканується індекс першого рівня.

Для проведення експерименту необхідно підготувати систему. Потрібно загрузити в систему необхідні словники:

– словник службових термінів: в даному експерименті використовується словник кількістю в 56 службових слів;

– словник, необхідний для проведення морфологічного аналізу: для російської мови такий словник існує кількістю в 1 200 000 слів, в даному експерименті з них використовується 18 357 слів. Цього вистачає для текстів, котрі беруть участь в експерименті;

– словники тематик: в даному випадку використовується чотири словники з наступних предметних областей: 1) біологія (8 786 слів); 2) правознавство (12 864 слів); 3) телекомунікації (10 272 слів); 4) електроніка (16 487 слів).

Тепер система повністю готова для побудови індексів. Для тестування було вибрано декілька текстів. Серед них є тексти як з однієї предметної області так і з інших предметних областей. Всі тексти вибрані для експерименту знаходяться в додатку.

Вводимо в строку запиту словосполучення «коммерческий кредит». На рисунку 4.3 зображено результат пошуку по даному запиту.

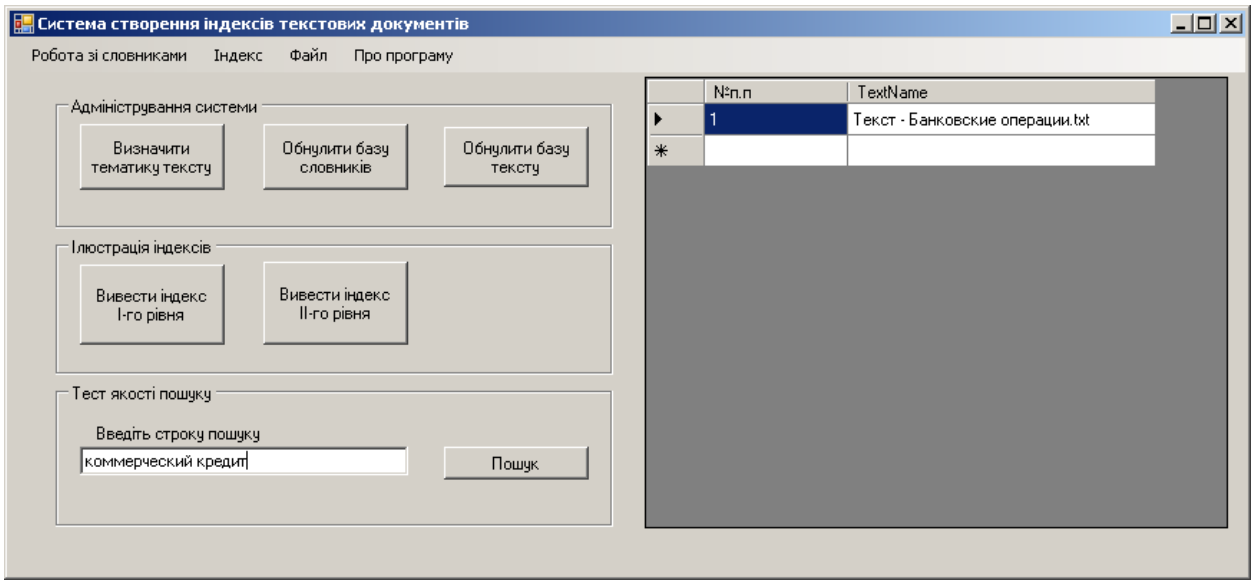


Рисунок 4.3 – Результати пошуку по запиту «коммерческий кредит»

Проаналізувавши досліджувані тексти, було виділено словосполучення «коммерческий кредит» в тексті «Банковские операции», в інших документах, що відносяться до класу «Правознавство» даного словосполучення не знайдено.

Отже, як видно з проведеного експерименту, на один і той же запит при класифікаційному індексуванні текстів, система видає всі документи, які належать до даного класу. Це є наслідком того, що слова запиту знаходяться в індексі даного класу, відповідно система видає всі тексти, котрі було до нього віднесено (Рисунок 4.2).

На відміну від класифікаційного, розроблений механізм індексування будує індивідуальний індекс для кожного тексту. Як результат, система вивела один текст «Банковские операции».

Тобто, на основі розробленого алгоритму індексування, точність пошуку вища.

4.2 Експеримент №2

В даному експерименті беруть участь два тексти з біології: «Природа генов» та «Эпистаз».

Нехай, запитом буде слово «рекомбинация».

На рисунках 4.4 та 4.5 зображено результат пошуку з використання класифікаційного та дворівневого механізмів індексування.

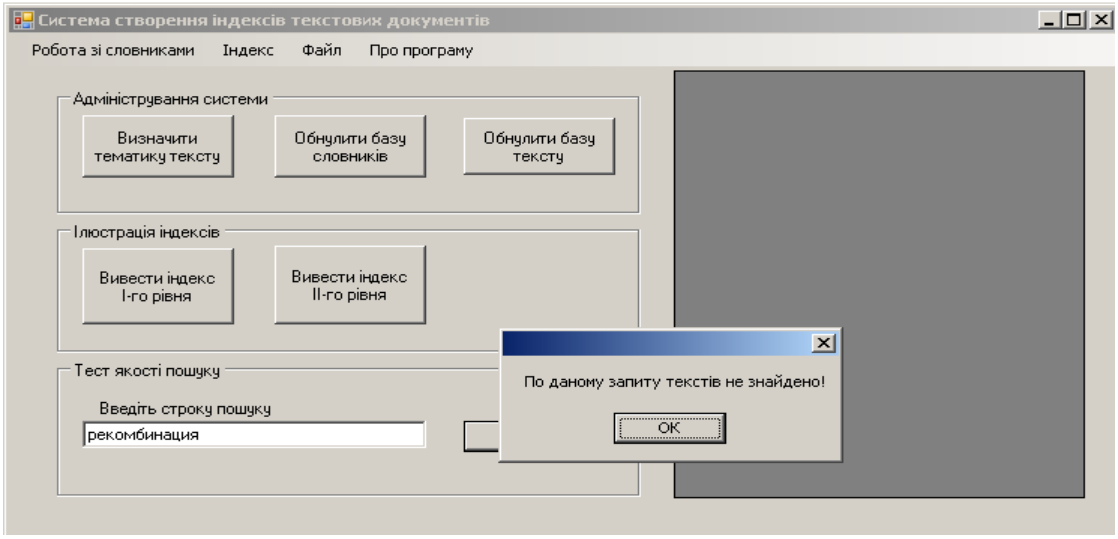


Рисунок 4.4 – Результат пошуку по запиту «рекомбинация» з класифікаційним алгоритмом індексування

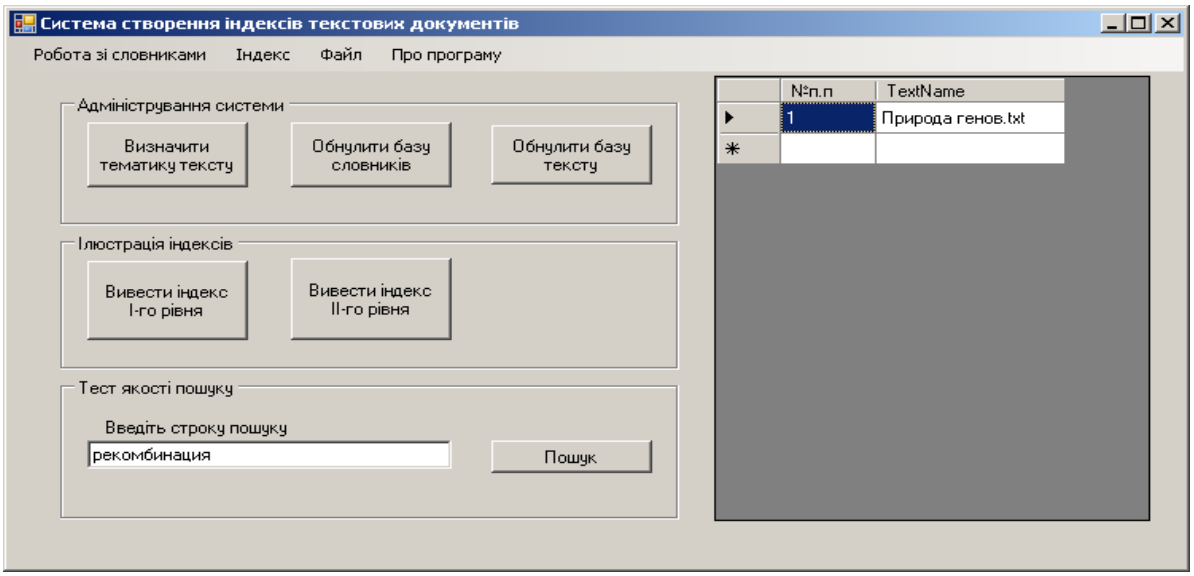


Рисунок 4.5 – Результат пошуку по запиту «рекомбинация» з дворівневим механізмом індексування

Проаналізувавши тексти, що брали участь в експерименті, було знайдено слово «рекомбинации» в документі «Природа генов.txt».

При обробці текстів дворівневим механізмом індексування, кожен з них під час створення індивідуального індексу проходить морфологічний аналіз, тобто враховуються всі словоформи кожного слова. На рисунку 4.5 бачимо, що система знайшла текст з таким словом, а система з класифікаційним індексуванням текстів дала нульовий результат пошуку.

4.3 Експеримент №3

Використовуються тексти, що брали участь в експерименті №2.

На рисунках 4.6 та 4.7 зображений результат пошуку текстів з запитом «полигенное наследование».

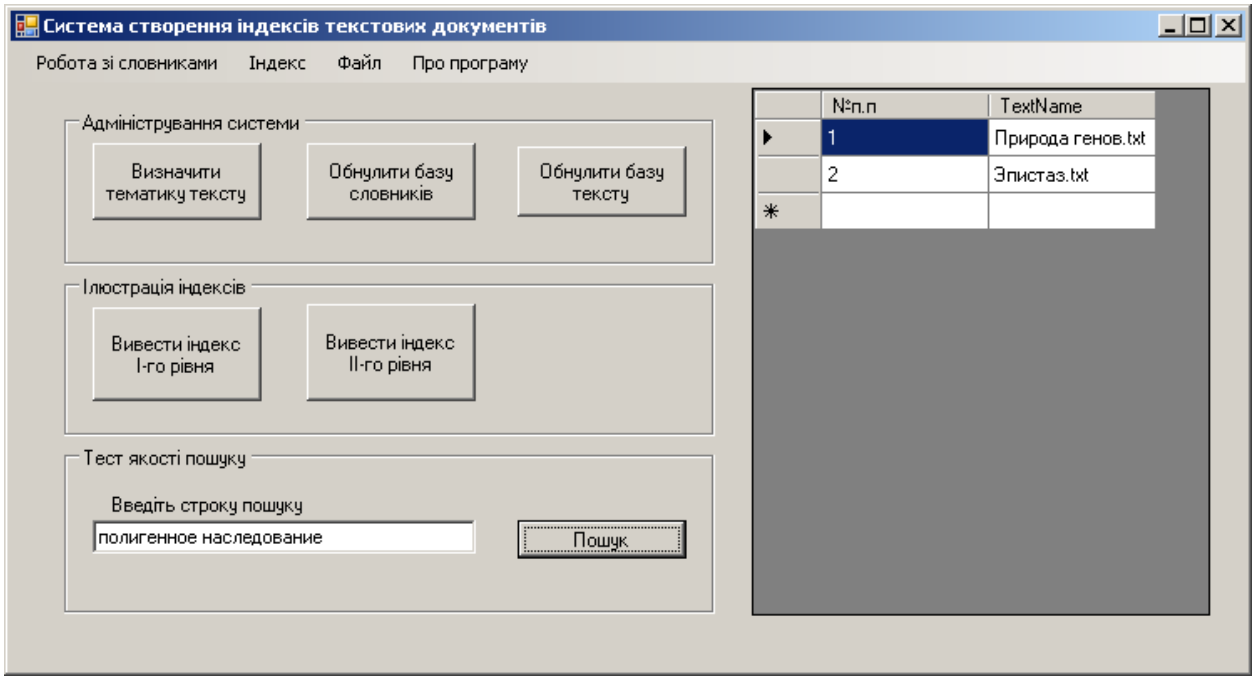


Рисунок 4.6 – Результат пошуку по запиту «полигенное наследование» в системі з класифікаційним алгоритмом індексуванням

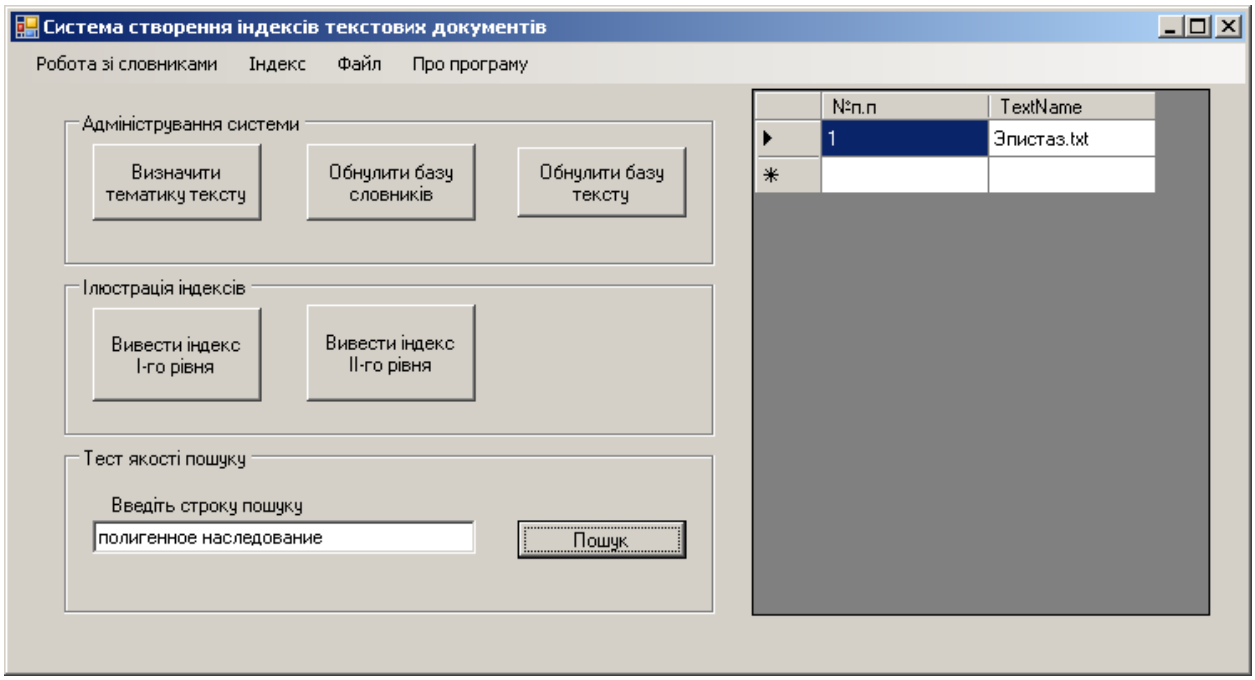


Рисунок 4.7 – Результат пошуку по запиту «полигенное наследование» в системі з класифікаційним алгоритмом індексуванням

Проаналізувавши тексти, словосполучення «полигенное наследование» знайдено в документі «Эпистаз.txt» і нічого про це не говорилося в «Природа генов.txt». Висновок аналогічний експерименту №1.

4.4 Експеримент №4

Ціллю даного експерименту являється аналіз швидкості пошуку запиту користувача в залежності від кількості текстів в базі. На рисунку 4.8 представлено вікно програми, де зображено повідомлення про швидкість обробки запиту.

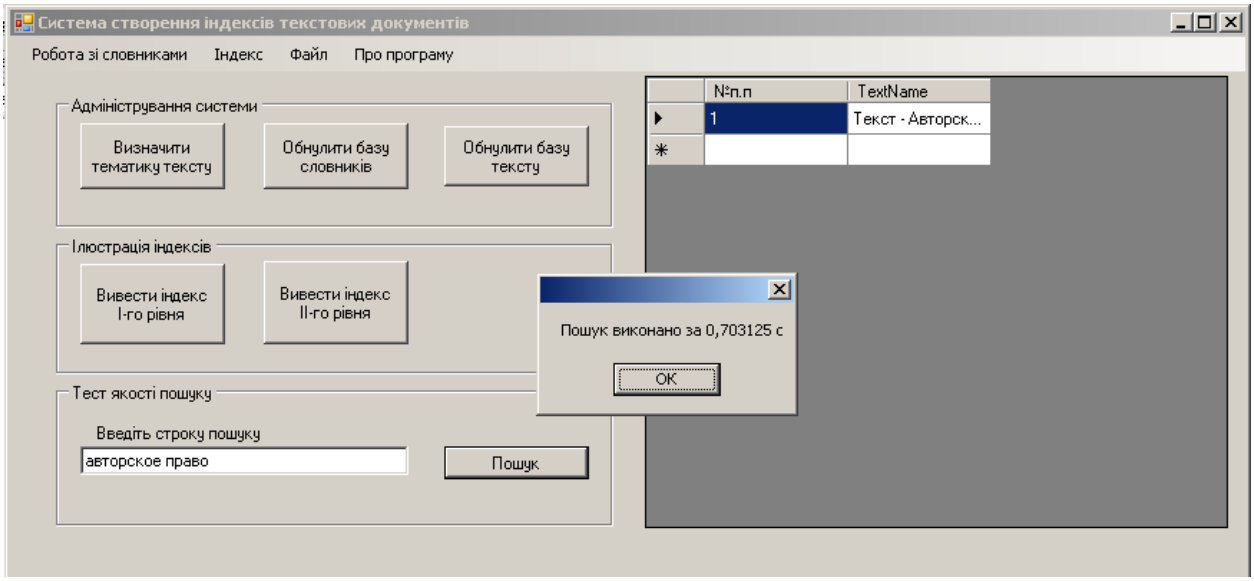


Рисунок 4.8 – Швидкість пошуку запиту в базі індексів

Результати експерименту представлені в таблиці 4.1.

Таблиця 4.1 – Таблиця швидкості пошуку строки запиту в залежності від кількості текстів в базі

Час пошуку, сек / Кількість текстів	Тест №1	Тест №2	Тест №3	Тест №4	Тест №5	Середнє арифметичне
40	0,7656	0,71875	0,73437	0,75	0,7656	0,746864
80	0,84375	0,8281	0,8125	0,84375	0,8281	0,83124
120	0,9843	0,9758	0,9843	1	0,9687	0,98262
160	1,2031	1,1875	1,2812	1,2031	1,2187	1,21872
200	1,484	1,4687	1,4687	1,4531	1,4687	1,46864
240	1,828	1,8593	1,8437	1,8593	1,8281	1,84368
260	2,3125	2,35937	2,3437	2,28125	2,29687	2,318738
320	2,7812	2,75	2,8593	2,875	2,75	2,8031
360	3,3593	3,4531	3,3437	3,4843	3,42187	3,412454
400	4,0625	4,125	4,0468	4,04687	4,15625	4,087484
520	6,75	6,8593	6,3906	6,4375	6,5625	6,59998
640	10,1562	10,3281	10,1718	9,9375	9,75	10,06873

Графік залежності швидкості пошуку строки запиту від кількості текстів в базі зображено на рисунку 4.9.

Рисунок 4.9 – Графік залежності швидкості обробки запиту від кількості індексів в базі з використання класифікаційного алгоритму індексування

4.5 Результати експерименту

Проаналізувавши виконані експерименти можна зробити наступні висновки.

В розглянутих вище алгоритмах індексування (класифікаційному та координатному) є безліч плюсів:

1) класифікаційне індексування:

– велика швидкість пошуку;

– малий розмір індексу;

2) координатне індексування:

– велика точність індексування;

– можливість створення чіткої системи підказок під час пошуку.

Але, присутні і суттєві недоліки, про які говорилося вище:

1) класифікаційне індексування:

– низький рівень точності індексу;

– присвоєння різним текстам спільного індексу;

2) координатне індексування:

– мала швидкість пошуку;

– складність врахування зв’язків між словами в тезаурусі.

Дворівневий механізм індексування являє собою схрещення цих двох алгоритмів. При чому, при його створенні враховувалися всі переваги цих систем, а недоліки ліквідовані за рахунок їх модифікації: морфологічного аналізу кожного слова при створенні індексу; створенні індивідуального індексу для кожного тексту; а також виконується статистичний аналіз словників предметних областей, не враховуючи зв’язків між ними.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Ідея проекту полягає в створенні компанії з дослідження та розробки технологій штучного інтелекту. Розглянемо зміст ідеї, можливі напрямки застосування, основні переваги, які зможе отримати користувач представлено у таблиці 5.1.

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Дослідження та розробка технологій індексування текстів бази знань	1. Технологічний консалтинг	оптимізація діяльності технічних підрозділів компаній на основі наявного досвіду, технологічних напрацювань
	2. Аутсорсинг	Вирішення питань з генерування та зберігання даних. Використання технологій задля впровадження систематизованого застосунку, що допомагає створювати та отримувати легкий доступ до контенту різного типу не використовуючи час програмістів. Розвиток та підтримка функціонування окремих ділянок системи.
	3. Розробка та патентування технологій	використання готових передових рішень

Даний проект відрізняється тим, що спеціалізується на певній предметній області.

На ринку наявні конкуренти, які надають схожий набір послуг, але вони не спеціалізуються на конкретній предметній області, що не дозволяє витрачати велику кількість ресурсів на дослідження технологій індексування та генерування даних, а отже якість цих послуг може бути нижчою.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 5.2):

Таблиця 5.2 - Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Дослідження та розробка технологій індексування текстів бази знань	Алгоритм перенесення знання спайкових мереж в штучні мережі	Вже розроблено	Технологія доступна
	Розробка системи для управління	Середовище розробки Visual Studio 2013, мова C#	Технологія доступна
		Середовище розробки Visual Studio 2013, мова	Технологія доступна

		C++	
Обрана технологія реалізації ідеї проекту: в якості середовища розробки було обрано Visual Studio та мову C# зважаючи на швидкість написання та універсальність мови; Для зберігання данних було вибрано базу даних Access через простоту використання та універсальність в структурі.			

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.3).

Таблиця 5.3 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	20000 за рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	70

Ринок є привабливий для входу та потребує новітніх алгоритмів пошуку найкращих, оптимальних методів вирішення задач.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 5.4).

Таблиця 5.4 - Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Розробка технологій індексування текстів бази знань	Потенційна група клієнтів є технологічні та консалтингові компанії	-потреба в продуті оптимізації -рішення які надають	-якісне надання послуг -технічна консультація та документація

		конкурентну перевагу підприємству	

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 5.5 та 5.6). Фактори в таблиці подаються в порядку зменшення значущості.

Таблиця 5.5 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	З’являються конкуренти	На ринку утворюються конкуренти з якіснішими послугами	-купівля компанії-конкурента -удосконалення якості продукту
2	Зменшиться попит	Ринок великий і динамічний - через це можливе зменшення попиту на рішення та використання технологій індексування текстів	-маркетинг -виготовлення універсальних програмних рішень

Таблиця 5.6 - Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Збільшиться попит	Збільшиться попит на новітні розробки систем індексування текстів	-підвищення ціни -виготовлення універсальних програмо-апаратних рішень -розширення ринку
2	Ріст іноземних інвестицій	Ріст інвестицій призведе до розширення	-розширення ринку

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таблиця 5.7).

Таблиця 5.7 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1.Вказати тип конкуренції - олігополія	Домінує мала кількість фірм	Пришвидшення розроблення унікального продукту
2. За рівнем конкурентної боротьби національний	Боротьба ведеться на національному ринку	Вихід на міжнародний ринок
3. За галузевою ознакою - внутрішньогалузева	Боротьба між товариствами галузі штучного інтелекту	Збільшення якості надання послуг
4. Конкуренція за	Різновиди однієї	Відсутня

видами товарів: - товарно-видова	категорії товару, які здатні задовольнити конкретне бажання покупця	
5. За характером конкурентних переваг - нецінова	Ключовим фактором конкурентної спроможності є експертиза в предметній області	Збільшення кількості наукових розробок
6. За інтенсивністю - не марочна	Не прив'язаний до певної марки, може використовувати будь-де	Співпраця з різними марками

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 5.8).

Таблиця 5.8 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Наведені в таблиці 5.2	Можуть з'явитись надаючи більш ширший спектр послуг	Відсутні	Клієнтам потрібні як готові рішення так і специфічні	Частково присутні
Висновки:	Конкуренти	Потенційні	Для	Можна	Простіші

	не мають достатньої експертизи	конкуренти можуть виконувати простіші замовлення	написання програмного забезпечення потрібне ліцензійне середовище	виграти на вирішенні специфічн их задач	рішення присутні на ринку

Робота на ринку можлива, попри наявність конкурентів, через досягнуту експертизу в області нейронних мереж конкуренти не здатні на виконання на стільки складних розробок; але простіші завдання конкуренти в змозі виконувати.

На основі аналізу конкуренції, проведеного в таблиця 5.8, а також із урахуванням характеристик ідеї проекту (таблиця 5.2), вимог споживачів до товару (таблиці 4.5) та факторів маркетингового середовища (таблиці 5.7 та 5.8) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею 4.10.

Таблиця 5.9 - Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Експертиза в предметній області	Експертиза в області дозволяє розв’язувати складні проблеми
2	Можливість розробки рішень для специфічних проблем	Компанія надає не лише послуги розробки, але й досліджень, тому вирішуються проблеми, для яких немає готових рішень на ринку
3	Спектр послуг	Спекрт послуг, які можна

		отримати при розробці

За визначеними факторами конкурентоспроможності (табл. 5.9) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 5.10). (С.П. – стартап проект, К.1 – Конкурент 1, К.2 – Конкурент 2)

Таблиця 5.10 - Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспромо жності	Бали (1-20)	Рейтинг товарів-конкурентів у порівнянні з … (Конкурент 1,2,3)						
			-3	-2	-1	0	+1	+2	+3
1	Експертиза в предметній області	20			К.2			К.1	С.П.
2	Можливість розробки рішень для специфічних проблем	20					К.1 К.2		С.П.
3	Спектр послуг	18		К.2				С.П.	К.1

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл.5.9).

Таблиця 5.11 - SWOT- аналіз стартап-проекту

Сильні сторони: експертиза в предметній області	Слабкі сторони: вузький спектр послуг
Можливості: збільшення попиту	Загрози: зменшення попиту, конкуренція

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (таблиця 4.9, аналіз потенційних конкурентів).

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таблиця. 5.12).

Таблиця 5.12 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Отримання основних клієнтів	середня	До 6 місяців
2	Отримання клієнтів з шаблонними задачами	висока	Необмежено
3	Розробка та продаж продуктів індексування текстів бази знань для вирішення тривіальних задач	висока	До одного року

Після аналізу зазначити обрану альтернативу.

Першою альтернативою є розробка та продаж продуктів системи індексування текстів для вирішення тривіальних задач, на який існує масовий попит.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 5.13).

Таблиця 5.13 - Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Компанії, яким у яких немає експертизи в області систем індексування текстів та потребують певних програмних рішень	Споживачі готові сприйняти продукт.	Попит є в цільовому сегменті	Конкуренція невелика	Вхід в сегмент не складатиме значних зусиль
Які цільові групи обрано: технологічні компанії				

За результатами аналізу потенційних груп споживачів (сегментів) обираються цільові групи, для яких пропонується товар, та визначається стратегія охоплення ринку:

- якщо компанія зосереджується на одному сегменті – вона обирає стратегію концентрованого маркетингу;
- якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – вона використовує стратегію диференційованого маркетингу;
- якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – вона використовує масовий маркетинг.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 4.15).

Таблиця 5.14 -Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Стратегія виклику лідера	Стратегія спеціалізації	Індивідуальний підхід до клієнта; краща якість відповідно до місця реалізації	Стратегія спеціалізації передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок. Мета тут полягає в задоволенні потреб вибраного

			цільового сегменту краще, ніж конкуренти.

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 8.16).

Таблиця 5.15 - Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Частково	Шукати нових та забирати споживачів у конкурентів	Ні	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 5.13), а також в залежності від обраної базової стратегії розвитку (табл. 5.14) та стратегії конкурентної поведінки (табл. 5.15) розробляється стратегія позиціонування (табл. 4.17). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.16 - Визначення стратегії позиціонування

Вимоги до товару цільової	Базова стратегія розвитку	Ключові конкурентоспроможні	Вибір асоціацій, які мають

аудиторії		позиції власного стартап-проекту	сформувати комплексну позицію власного проекту (три ключових)
Вирішення нетипових задач систем індексування текстів бази знань	Стратегія спеціалізації	Складні задачі Дослідження нових методів Індивідуальні рішення	глибока експертиза, складні задачі, сучасні розробки

Результатом виконання підрозділу є узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії на ринку.

5.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.17 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі,

			що потрібно створити
1	вирішення простих задач за допомогою методів індексування текстів	шаблонні та коробочні рішення	добра якість за доступну ціну
2	вирішення складних задач за допомогою методів систем індексування текстів бази знань	комплексний та глибокий підхід до вирішення проблеми	ефективне рішення складних задач

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 5.18)

Таблиця 5.18 - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Надання сервісів в сфері індексування та генерування текстів		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	-		
	Якість: стандарти, нормативи, параметри тестування тощо		
	Технологічний консалтинг, аутсорсинг та розроблені рішення в сфері систем індексування текстів		

	Марка: назва компанії
III. Товар із підкріпленням	Готовий продукт
	Технічна підтримка
Патенти	

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 5.19 - Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
50 тис. грн	75 тис. грн	500 тис. грн	10 тис. грн – 100 тис. грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 5.20):

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 5.20 - Формування системи збуту

Специфіка	Функції збуту,	Глибина каналу	Оптимальна

закупівельної поведінки цільових клієнтів	які має виконувати постачальник товару	збуту	система збуту
Знаходження на сайті послуг, вибір конкретних послуг, обговорення завдання, оплата	-	Виробник-споживач	Web-сайт

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 5.21 - Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Знають, які саме послуги треба для вирішення задач	Веб-сайт, телефон, зустрічі	Підтримка, індивідуальний підхід	Донесення переваг до клієнтів	Вирішення задач із індексуванням текстів різної складності

Результатом пункту 5 є ринкова (маркетингова) програма, що включає в себе концепції послуг, збуту, просування та попередній аналіз можливостей

ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки: Є можливість ринкової комерціалізації проекту. Попит на послуги присутні, попри наявність продуктів-аналогів. Розроблений проект має свої переваги над конкурентами у вигляді співвідношення ціна - якість. З огляду на потенційні групи клієнтів є перспективи для входження на ринок. Отже конкурентоспроможність послуг висока. В якості базової стратегії розвитку обрана стратегія специфікації. В якості альтернативи впровадження доцільно обрати комбінацію з іншими методами аналізу зображення та розміщення реклами послуг.

ВИСНОВКИ

Основною задачею систем управління знаннями є якісне їх зберігання та легкий доступ до них.

Було проведено аналіз існуючих рішень не лише в області управління знаннями, а й розглянуто ряд систем управління документообігом, пошукові системи, та різні алгоритми індексування, які в них застосовуються.

При розробці механізму індексування текстів було поставлено певні вимоги:

– простота реалізації – від цього буде залежати ціна продукту;

– точність індексування – від точності індексування тексту, залежить якість подальшого його використання в системі управління знаннями;

– смисловий пошук – виявлення семантики термів;

– відкрита архітектура – розширення можливостей модулю без затрати на це великих ресурсів;

– багатомовність – мається на увазі незалежність роботи механізму від вибраної мови;

– висока швидкість пошуку;

– висока швидкість оброблення текстових документів;

– простота інтерфейсу користувача.

Описана в роботі система, повністю задовольняє даним вимогам, що підтверджують результати експерименту.

В основу механізму були покладені два основні алгоритми індексування: класифікаційне та координатне, та модифікована латентно-семантична математична модель.

Дворівневий механізм індексування являє собою схрещення цих двох алгоритмів. При чому, при його створенні враховувалися їх переваги, а недоліки ліквідовано за рахунок модифікації: морфологічного аналізу

кожного слова при створенні індексу; створенні індивідуального індексу для кожного тексту.

Програмний продукт розроблено в середовищі програмування MS Visual Studio 2017 Community Edition, на мові програмування C# під платформою Microsoft .NET Framework 4.5. База даних реалізована в пакеті Microsoft Access.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Knowledge Management, Content Management, Sharepoint – KMWorld Magazine [Електронний ресурс]. – March, 2016. – Режим доступу: http://www.kmworld.com/Issues/Archives. – Назва з екрану. – 10.10.2018 р.

2. К-технологии [Електронний ресурс]. – Режим доступу: www.k-technology.ru/knowledge/. – Назва з екрану. – 12.10.2018 р.

3. [Management.com.ua] Корпоративные знания: как ими управлять? [Електронний ресурс]. – Режим доступу: www.management.com.ua/ims/ims108.html. – Назва з екрану. – 12.10.2018 р.

4. Knowledge Management Magazine [Электронный ресурс]. – Режим доступу: http://www.kmmagazine.com/. – Назва з екрану. – 13.10.2018 р.

5. Infosphere Information Server [Електронний ресурс]. – Режим доступу: https://www.ibm.com/analytics/information-server. – Назва з екрану. – 10.10.2018р.

6. Livelink | OpenText [Електронний ресурс]. – Режим доступу: https://www.opentext.com/products-and-solutions/products/opentext-product-offerings-catalog/rebranded-products/livelink-is-now-part-of-the-opentext-ecm-suite. – Назва з екрану. – 14.10.2018 р.

7. Электронный документооборот «Логика «СЭД» [Електронний ресурс]. – Режим доступу: http://ecm.blogic20.ru/logikaecm/docflow-solutions. – Назва з екрану. – 15.10.2018 р.

8. Система электронного документооборота DocVision [Електронний ресурс]. – Режим доступу: https://docsvision.com/. – Назва з екрану. – 15.10.2018 р.

9. Электронный документооборот Directum [Електронний ресурс]. – Режим доступу: https://www.directum.ru/. – Назва з екрану. – 15.10.2018 р.

10. Платформа для создания систем управления знаниями [Електронний ресурс]. – Режим доступу: http://soft.neurok.ru/products/semantic.shtml. – Назва з екрану. – 15.10.2018 р.

11. diskMETA-Lite full text search engine [Електронний ресурс]. – Режим доступу: http://www.diskmeta.com/en/dmlite/. – Назва з екрану. – 15.10.2018 р.

12. MTSearch – Пролинг ОФИС [Електронний ресурс]. – Режим доступу: http://prolingoffice.com/product/mtsearch. – Назва з екрану. – 15.10.2018 р.

13. Пошукові системи склад, функції, принцип роботи [Електронний ресурс]. – Режим доступу: https://pandia.ru/text/79/494/19763.php. – Назва з екрану. – 15.10.2018 р.

14. Baeza-Yates R. Modern information retrieval / R. Baeza-Yates, B. Ribeiro-Neto. – 1999. – 513 p.

15. Зупарова Л. В. Библиотечная обработка документа: под редакцией професора Ю. Н. Столярова / Л.В. Зупарова, Т.Д. Зайцева, Л.І. Сазонова. – 2003. – 208 с.

16. Индексирование текстов поисковыми роботами [Електронний ресурс]. – Режим доступу: http://www.onlinehomebusiness.ru/articles_858.html. – Назва з екрану. – 20.10.2018 р.

17. Голуб Дж. Матричные вычисления / Дж. Голуб, Ч. Ван Лоун. – М.: Мир, 1999. – 548 с.

18. Landauer T. An introduction to latent semantic analysis / T. Landauer, P. Foltz, D. Laham // Discourse Processes, 25. – p. 259-284.

19. Добринин В.Ю. Теория информационно-логических систем / В.Ю. Добринин // Информационный поиск. – Видавництво СпбГУ, 2012. – 38 с.

20. Троелсен Е. C# и платформа .NET. – Питер, 2016. – 796 стр.

ДОДАТОК А

Код програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Data.OleDb;

namespace TextIndeks
{
    public partial class Form1 : Form
    {
        DataRow row;
        DataTable tableWord = new DataTable("TextWord");
        DataTable tIndexText = new DataTable("TableIndex");
        List<DataTable> lstTablesSlovn = new List<DataTable>();
        DataTable tQuery = new DataTable("Query");
        DataTable tableSlov2 = new DataTable("Slovnuk2");
        DataTable tableIndex = new DataTable("Index");

        DataColumn idWord = new DataColumn();
        DataColumn Word = new DataColumn();

        DataColumn idIndex = new DataColumn();
        DataColumn TextIndex = new DataColumn();
        DataColumn kilWordBaz = new DataColumn();

        DataColumn idQ = new DataColumn();
        DataColumn WordQ = new DataColumn();

        DataColumn idWordSlov2 = new DataColumn();
        DataColumn WordSlov2 = new DataColumn();

        OleDbDataAdapter da = new OleDbDataAdapter();
        OleDbConnection cn = new OleDbConnection();

        OpenFileDialog op = new OpenFileDialog();
        string str;
        static public int i = 0;
        public int kilW = 0;
        public int kilWS1 = 0;
        public int kilWS2 = 0;
        public int kilS = 0;
        static public int kilBazW = 0;
        public int flagS = 0;
        public int kilSlov1 = 0;
        public int kilSlov2 = 0;
        public int flag = 0;

        public Form1()
        {
            InitializeComponent();

            idWord.DataType = Type.GetType("System.Int32");
            idWord.ColumnName = "№п.п";
            idWord.AutoIncrement = true;
            idWord.AutoIncrementSeed = 1;
            idWord.AutoIncrementStep = 1;

            Word.DataType = Type.GetType("System.String");
            Word.Caption = "Word";
            Word.ColumnName = "Word";

            tableWord.Columns.Add(idWord);
            tableWord.Columns.Add(Word);

            kilWordBaz.DataType = Type.GetType("System.Int32");
            kilWordBaz.ColumnName = "Кількість слова в тексті";
            kilWordBaz.Caption = "kilWordBaz";
```

```
    idIndex.DataType = Type.GetType("System.Int32");
    idIndex.ColumnName = "№п.п";
    idIndex.AutoIncrement = true;
    idIndex.AutoIncrementSeed = 1;
    idIndex.AutoIncrementStep = 1;

    TextIndex.DataType = Type.GetType("System.String");
    TextIndex.Caption = "TextName";
    TextIndex.ColumnName = "TextName";

    tIndexText.Columns.Add(idIndex);
    tIndexText.Columns.Add(TextIndex);
    //tableBaz.Columns.Add(kilWordBaz);

    idQ.DataType = Type.GetType("System.Int32");
    idQ.ColumnName = "№п.п";
    idQ.AutoIncrement = true;
    idQ.AutoIncrementSeed = 1;
    idQ.AutoIncrementStep = 1;

    WordQ.DataType = Type.GetType("System.String");
    WordQ.Caption = "WordQuery";
    WordQ.ColumnName = "WordQuery";


    idWordSlov2.DataType = Type.GetType("System.Int32");
    idWordSlov2.ColumnName = "№п.п";
    idWordSlov2.AutoIncrement = true;
    idWordSlov2.AutoIncrementSeed = 1;
    idWordSlov2.AutoIncrementStep = 1;

    WordSlov2.DataType = Type.GetType("System.String");
    WordSlov2.Caption = "WordSlovnuk2";
    WordSlov2.ColumnName = "WordSlovnuk2";

    idIndex.DataType = Type.GetType("System.Int32");
    idIndex.ColumnName = "№п.п";
    idIndex.AutoIncrement = true;
    idIndex.AutoIncrementSeed = 1;
    idIndex.AutoIncrementStep = 1;


    tQuery.Columns.Add(idQ);
    tQuery.Columns.Add(WordQ);
}

private DataTable NewSlov(string slovnName)
{
    DataTable tSlov = new DataTable(slovnName);

    DataColumn idWordNew = new DataColumn();
    DataColumn WordNew = new DataColumn();

    idWordNew.DataType = Type.GetType("System.Int32");
    idWordNew.ColumnName = "№п.п";
    idWordNew.AutoIncrement = true;
    idWordNew.AutoIncrementSeed = 1;
    idWordNew.AutoIncrementStep = 1;

    WordNew.DataType = Type.GetType("System.String");
    WordNew.Caption = "WordSlovnuk";
    WordNew.ColumnName = "WordSlovnuk";

    tSlov.Columns.Add(idWordNew);
    tSlov.Columns.Add(WordNew);

    lstTablesSlovn.Add(tSlov);
    return tSlov;
}

private void відкритиФайлToolStripMenuItem_Click(object sender, EventArgs e)
{

    bool f;
    int idF = 0;
    Stream myStream;
    bool flag1 = false;
    string s1 = "";
    string s3 = "";
```

```
            op.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";

            if (op.ShowDialog() == DialogResult.OK)
            {
                string str = "";
                if ((myStream = op.OpenFile()) != null)
                {
                    string TextName = op.SafeFileName;

                    FileInfo tf = new FileInfo(op.FileName);
                    Text text = TextMng.Instance.GetItemByName(tf.Name);


                    if (text == null)
                    {
                        Text newT = TextMng.Instance.NewItem();
                        newT.name = tf.Name;
                        TextMng.Instance.Save(newT);
                        int idText = TextMng.Instance.GetIDByName(tf.Name);



                        using (TextReader strRd = new StreamReader(myStream))
                        {
                            string strW = "";
                            while ((strW = strRd.ReadLine()) != null)
                            {
                                string[] wrds = strW.ToLower().Split(new char[] { ' ', ',', '.',
'(', ')', ';', '"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                                foreach (string s in wrds)
                                {

                                    StreamReader sr =
File.OpenText(@"D:\08.05.22\Slovari\МорфолСловарь\морфСлов.txt");
                                    while ((s1 = sr.ReadLine()) != null)
                                    {
                                        flag1 = false;
                                        string[] w = s1.ToLower().Split(new char[] { ' ', ',',
'.', '(', ')', ';', '"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                                        foreach (string s2 in w)
                                        {
                                            s3 = s2;
                                            break;
                                        }

                                        foreach (string s4 in w)
                                        {
                                            if (s4 == s)
                                            {
                                                idF = SlugWordMng.Instance.GetIdByName(s);
                                                if (idF == 0)
                                                {
                                                    WordText newW =
WordTextMng.Instance.NewWord();

                                                    newW.WordName = s3;
                                                    newW.IDText =
TextMng.Instance.GetIDByName(tf.Name);

                                                    WordTextMng.Instance.Save(newW);
                                                }
                                                idF = 0;
                                                flag1 = true;
                                                break;
                                            }
                                        }
                                        if (flag1 == true)
                                        {
                                            break;
                                        }
                                    }


                                }
                            }
                        }
                        myStream.Close();
                        WordTextMng.Instance.ToFile();
```

```
                        string strQuery = @"SELECT tWordText.WordText AS Word,
Count(tWordText.WordText) AS CountWord
FROM (tWordText INNER JOIN tWordDic ON tWordText.WordText = tWordDic.Word) WHERE tWordText.IDText
=" + idText.ToString() +
                    " GROUP BY tWordText.WordText ORDER BY Count(tWordText.WordText) DESC;";

                        OleDbConnection cn = new OleDbConnection(Settings.StrConnection);
                        cn.Open();
                        OleDbCommand cmd = new OleDbCommand(strQuery, cn);
                        OleDbDataReader dr = cmd.ExecuteReader();
                        while (dr.Read())
                        {
                            string word = Convert.ToString(dr["Word"]);
                            int idWord = WordTextMng.Instance.GetIDByExistName(word);
                            int kil = Convert.ToInt32(dr["CountWord"]);

                            string strSelect = @"SELECT tDictionary.ID AS idDic FROM
tDictionary;";
                            OleDbCommand cmdS = new OleDbCommand(strSelect, cn);
                            OleDbDataReader dr1 = cmdS.ExecuteReader();
                            while (dr1.Read())
                            {
                                int idD = Convert.ToInt32(dr1["idDic"]);
                                DicContentMng.Instance.setWordTextDicContent(idD, idWord, idText,
kil, word);
                            }
                        }
                        cn.Close();
                    }
                }

                //gridBazis.Visible = true;
                //gridBazis.DataSource = tableWord;
            }

            WordTextMng.Instance.TextTopic();


            int idDic = 0;
            string dicName = "";
            string textName = "";
            string word1 = "";
            string wordText = "";
            bool flag = false;

            idDic = TextSubjectMng.Instance.GetIdDic();
            textName = TextMng.Instance.GetName();

            TextIndex ti = TextIndexMng.Instance.NewItem();
            ti.Name = textName;
            TextIndexMng.Instance.Save(ti);

            FileInfo f2 = new FileInfo(@"D:\08.05.22\Indeks\I\" + textName + "");
            StreamWriter write = f2.CreateText();

            string strQ = @"SELECT tWordTextDicContent.Word AS W FROM tWordTextDicContent
WHERE tWordTextDicContent.IDDictionary =" + idDic.ToString() + ";";
            string strQ1 = @"SELECT tWordText.WordText AS WT FROM tWordText;";

            OleDbConnection con = new OleDbConnection(Settings.StrConnection);
            con.Open();

            OleDbCommand odc = new OleDbCommand(strQ1, con);
            OleDbDataReader read = odc.ExecuteReader();

            while (read.Read())
            {
                wordText = Convert.ToString(read["WT"]);

                OleDbCommand com = new OleDbCommand(strQ, con);
                OleDbDataReader dbr = com.ExecuteReader();

                while (dbr.Read())
                {
                    word1 = Convert.ToString(dbr["W"]);
                    if (wordText == word1)
                    {
                        write.WriteLine(word1);
                        //flag = true;
```

```
                    break;

                }
            }
        }
        write.Close();

        textName = "";
        string str1 = "";
        idDic = 0;
        s1 = "";
       string s7 = "";

        flag1 = false;
        bool flag2 = false;
        string word3 = "";
        string word2 = "";

        idDic = TextSubjectMng.Instance.GetIdDic();
        textName = TextMng.Instance.GetName();

        TextIndex ti1 = TextIndexMng.Instance.NewItem();
        ti1.Name = textName;
        TextIndexMng.Instance.Save(ti1);

        OleDbConnection c = new OleDbConnection(Settings.StrConnection);
        c.Open();

        idDic = TextSubjectMng.Instance.GetIdDic();

        textName = TextMng.Instance.GetName();
        FileInfo f1 = new FileInfo(@"D:\18.10.22\Indeks\II\" + textName + "");
        StreamWriter write1 = f1.CreateText();

        StreamReader sr1 = File.OpenText(@"D:\18.10.22\CurText\" + textName + "");
        while ((str1 = sr1.ReadLine()) != null)
        {
            row = tableWord.NewRow();
            tableWord.Rows.Add(row);
            row["Word"] = str1;
        }

        DataRowCollection rowsIn = tableWord.Rows;
        for (int i = 0; i < tableWord.Rows.Count - 1; i++)
        {
            DataRow r = rowsIn[i];
            DataRow r1 = rowsIn[i + 1];
            s1 = r["Word"].ToString();
            s7 = r1["Word"].ToString();

            string strQuery1 = @"SELECT tDicContent.Word AS W1 FROM tDicContent
WHERE tDicContent.IDDictionary =" + idDic.ToString() + ";";

            string strQuery2 = @"SELECT tDicContent.Word AS W2 FROM tDicContent
WHERE tDicContent.IDDictionary =" + idDic.ToString() + ";";

            OleDbCommand cm1 = new OleDbCommand(strQuery1, c);
            OleDbDataReader dbr1 = cm1.ExecuteReader();
            while (dbr1.Read())
            {
                word3 = Convert.ToString(dbr1["W1"]);
                if (word3 == s1)
                {
                    flag1 = true;
                }
            }

            OleDbCommand cm2 = new OleDbCommand(strQuery2, c);
            OleDbDataReader dbr2 = cm2.ExecuteReader();
            while (dbr2.Read())
            {
                word2 = Convert.ToString(dbr2["W2"]);
                if (word2 == s7)
                {
                    flag2 = true;
                }
            }

            if ((flag1 == true) && (flag2 == true))
```

```
        {
            write1.WriteLine(s1 + " " + s7);
        }

        flag1 = false;
        flag2 = false;

    }
    //gridBazis.Visible = true;
    //gridBazis.DataSource = tableWord;
    write1.Close();
    c.Close();
}
private void AddWord(string str)
{
    str = RemSymvol(str);
    if (tableWord.Rows.Count <= kilW)
    {
        row = tableWord.NewRow();
        tableWord.Rows.Add(row);
    }
    row = tableWord.Rows[kilW];
    str = str.ToLower();
    row["Word"] = str;
    kilW++;
}

private void AddWordSlov2(string slov2)
{
    slov2 = RemSymvol(slov2);
    if (tableSlov2.Rows.Count <= kilWS2)
    {
        row = tableSlov2.NewRow();
        tableSlov2.Rows.Add(row);
    }
    row = tableSlov2.Rows[kilWS2];
    slov2 = slov2.ToLower();
    row["WordSlovnuk2"] = slov2;
    kilWS2++;
}


private string RemSymvol(string str)
{
    return str;

}

private void вихідToolStripMenuItem_Click_1(object sender, EventArgs e)
{
    Close();
}

private void відкритиСловникТематикToolStripMenuItem_Click(object sender, EventArgs e)
{

    int countInDictionary = 0;
    int countAdded = 0;
    int countWordAdded = 0;
    int idWordSlug = 0;
    string s2 = "";
    string s4 = "";
    bool flag1 = false;

    DateTime dateStart = DateTime.Now;

    Stream myStream;
    op.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";

    if (op.ShowDialog() == DialogResult.OK)
    {
        bool flag;
        str = "";
        if ((myStream = op.OpenFile()) != null)
        {
            string fileName = op.SafeFileName;

            FileInfo f = new FileInfo(op.FileName);
            Dictionary dic = DictionaryMng.Instance.GetItemByName(f.Name);
```

```
                    if (dic == null)
                    {
                        Dictionary newd = DictionaryMng.Instance.NewItem();
                        newd.Name = f.Name;
                        DictionaryMng.Instance.Save(newd);
                        int idDic = DictionaryMng.Instance.GetIDByName(f.Name);

                        using (TextReader strRd = new StreamReader(myStream))
                        {

                            string strWS1 = "";
                            while ((strWS1 = strRd.ReadLine()) != null)
                            {
                                string[] wrds1 = strWS1.ToLower().Split(new char[] { ' ', ',',
'.', '(', ')', ';', '"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                                foreach (string s1 in wrds1)
                                {

                                    StreamReader sr =
File.OpenText(@"D:\Studi\!!!ucheba\XI_semestr\Magisters'ka\22.10.18_v1\22.10.18\Slovari\МорфолСло
варь\морф.txt");
                                    while ((s2 = sr.ReadLine()) != null)
                                    {
                                        flag1 = false;
                                        string[] w1 = s2.ToLower().Split(new char[] { ' ', ',',
'.', '(', ')', ';', '"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                                        foreach (string s3 in w1)
                                        {
                                            s4 = s3;
                                            break;
                                        }

                                        foreach (string s5 in w1)
                                        {
                                            if (s5 == s1)
                                            {
                                                flag = DicContentMng.Instance.FindItem(idDic,
s1);
                                                idWordSlug =
SlugWordMng.Instance.GetIdByName(s1);

                                                WordDic w =
WordDicMng.Instance.GetItemByName(s1);
                                                if (w == null)
                                                {
                                                    WordDic newW = WordDicMng.Instance.NewWord();
                                                    newW.WordName = s4;
                                                    WordDicMng.Instance.Save(newW);
                                                    countWordAdded++;

                                                    if ((flag == false) && (idWordSlug == 0))
                                                    {
                                                        DicContent newCon =
DicContentMng.Instance.NewItem();
                                                        newCon.IDDictionary =
DictionaryMng.Instance.GetIDByName(f.Name);

                                                        //newCon.IDWord =
WordMng.Instance.GetIDByNewName(s1);
                                                        newCon.WordName = s4;
                                                        newCon.IDWord =
WordDicMng.Instance.GetIDByExistName(s4);
                                                        DicContentMng.Instance.Save(newCon);
                                                        countInDictionary++;
                                                        countAdded++;
                                                    }

                                                }
                                                else
                                                {


                                                    if ((flag == false) && (idWordSlug == 0))
                                                    {
                                                        DicContent newCon =
DicContentMng.Instance.NewItem();
                                                        newCon.IDDictionary =
DictionaryMng.Instance.GetIDByName(f.Name);
```

```
                                                    newCon.WordName = s4;
                                                    newCon.IDWord =
WordDicMng.Instance.GetIDByExistName(s4);
                                                    DicContentMng.Instance.Save(newCon);
                                                    countInDictionary++;
                                                    countAdded++;
                                                }
                                            }
                                            flag1 = true;
                                            break;
                                        }
                                    }
                                    if (flag1 == true)
                                    {
                                        break;
                                    }
                                }
                            }
                        }
                        myStream.Close();
                    }
                }

                DateTime dateEnd = DateTime.Now;
                TimeSpan t = dateEnd.Subtract(dateStart);
                MessageBox.Show("Дані оброблено за " + t.TotalSeconds + " с" +
Environment.NewLine
                                + "Проаналізовано " + countInDictionary + " записів "
                                + Environment.NewLine + "Додано " + countWordAdded + "нових слів"
                                + Environment.NewLine + "Додано " + countAdded + " слів в
довідник");

            }
        }

        private void btnTemTekst_Click(object sender, EventArgs e)
        {

            WordTextMng.Instance.TextTopic();

        }

        private void btnClearDB_Click(object sender, EventArgs e)
        {
            DicContentMng.Instance.ClearTable();
            DictionaryMng.Instance.ClearTable();
            WordDicMng.Instance.ClearTable();
        }

        private void btnClearText_Click(object sender, EventArgs e)
        {
            WordTextMng.Instance.ClearTable();
            WordTextDicContentMng.Instance.ClearTable();
            TextMng.Instance.ClearTable();
            TextSubjectMng.Instance.ClearTable();
            DicWithoutSlugWordMng.Instance.ClearTable();
        }

        private void загрузитиСловникСоюзівToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Stream myStream;
            op.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";

            if (op.ShowDialog() == DialogResult.OK)
            {
                str = "";
                if ((myStream = op.OpenFile()) != null)
                {
                    using (TextReader strRd = new StreamReader(myStream))
                    {
                        string strW = "";
                        while ((strW = strRd.ReadLine()) != null)
                        {
                            SlugWord sw = SlugWordMng.Instance.NewItem();
                            sw.Name = strW;
                            SlugWordMng.Instance.Save(sw);
                        }
                    }
```

```
                myStream.Close();
            }
        }
     }

    private void текстиToolStripMenuItem_Click(object sender, EventArgs e)
    {

        //WordTextMng.Instance.WithoutSlugWord();
    }

    private void словникToolStripMenuItem_Click(object sender, EventArgs e)
    {
        string nameDic = "";
        int idDic = 0;
        string WordDic = "";
        int idSlug = 0;

        nameDic = TextSubjectMng.Instance.GetNameSlov();
        idDic = DictionaryMng.Instance.GetIDByNameDic(nameDic);

        string strQ = @"SELECT tDicContent.Word as W FROM tDicContent WHERE
tDicContent.IDDictionary = " + idDic.ToString() + ";";

        OleDbConnection cn1 = new OleDbConnection(Settings.StrConnection);
        cn1.Open();
        OleDbCommand odc = new OleDbCommand(strQ, cn1);
        OleDbDataReader odr = odc.ExecuteReader();
        while (odr.Read())
        {
            WordDic = Convert.ToString(odr["W"]);
            idSlug = SlugWordMng.Instance.GetIdByName(WordDic);
            if (idSlug == 0)
            {
                DicWithoutSlugWord dwsw = DicWithoutSlugWordMng.Instance.NewItem();
                dwsw.name = WordDic;
                DicWithoutSlugWordMng.Instance.Save(dwsw);
            }
            idSlug = 0;
        }
        cn.Close();
    }

    private void індекс2гоРівняToolStripMenuItem_Click(object sender, EventArgs e)
    {
        string textName = "";
        string str = "";
        int idDic = 0;
        string s1 = "";
        string s2 = "";

        bool flag1 = false;
        bool flag2 = false;
        string word1 = "";
        string word2 = "";

        idDic = TextSubjectMng.Instance.GetIdDic();
        textName = TextMng.Instance.GetName();

        TextIndex ti = TextIndexMng.Instance.NewItem();
        ti.Name = textName;
        TextIndexMng.Instance.Save(ti);

        OleDbConnection c = new OleDbConnection(Settings.StrConnection);
        c.Open();

        idDic = TextSubjectMng.Instance.GetIdDic();

        textName = TextMng.Instance.GetName();
        FileInfo f = new FileInfo(@"D:\18.10.22\Indeks\II\" + textName + "");
        StreamWriter write = f.CreateText();

        StreamReader sr = File.OpenText(@"D:\18.10.22\CurText\" + textName + "");
        while ((str = sr.ReadLine()) != null)
        {
            row = tableWord.NewRow();
            tableWord.Rows.Add(row);
            row["Word"] = str;
        }
```

```
            DataRowCollection rowsIn = tableWord.Rows;
            for (int i = 0; i < tableWord.Rows.Count - 1; i++)
            {
                DataRow r = rowsIn[i];
                DataRow r1 = rowsIn[i + 1];
                s1 = r["Word"].ToString();
                s2 = r1["Word"].ToString();

                string strQuery1 = @"SELECT tDicContent.Word AS W1 FROM tDicContent
WHERE tDicContent.IDDictionary =" + idDic.ToString() + ";";

                string strQuery2 = @"SELECT tDicContent.Word AS W2 FROM tDicContent
WHERE tDicContent.IDDictionary =" + idDic.ToString() + ";";

                OleDbCommand cm1 = new OleDbCommand(strQuery1, c);
                OleDbDataReader dbr1 = cm1.ExecuteReader();
                while (dbr1.Read())
                {
                    word1 = Convert.ToString(dbr1["W1"]);
                    if (word1 == s1)
                    {
                        flag1 = true;
                    }
                }
                OleDbCommand cm2 = new OleDbCommand(strQuery2, c);
                OleDbDataReader dbr2 = cm2.ExecuteReader();
                while (dbr2.Read())
                {
                    word2 = Convert.ToString(dbr2["W2"]);
                    if (word2 == s2)
                    {
                        flag2 = true;
                    }
                }
                if ((flag1 == true) && (flag2 == true))
                {
                    write.WriteLine(s1 + " " + s2);
                }

                flag1 = false;
                flag2 = false;
            }
            //gridBazis.Visible = true;
            //gridBazis.DataSource = tableWord;
            write.Close();
            c.Close();
        }

        private void індекс1гоРівняToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int idDic = 0;
            string dicName = "";
            string textName = "";
            string word = "";
            string wordText = "";
            bool flag = false;

            idDic = TextSubjectMng.Instance.GetIdDic();
            textName = TextMng.Instance.GetName();

            TextIndex ti = TextIndexMng.Instance.NewItem();
            ti.Name = textName;
            TextIndexMng.Instance.Save(ti);

            FileInfo f = new FileInfo(@"D:\18.10.22\Indeks\I\" + textName + "");
            StreamWriter write = f.CreateText();


            string strQ = @"SELECT tWordTextDicContent.Word AS W FROM tWordTextDicContent
WHERE tWordTextDicContent.IDDictionary =" + idDic.ToString() +";";
            string strQ1 = @"SELECT tWordText.WordText AS WT FROM tWordText;";

            OleDbConnection con = new OleDbConnection(Settings.StrConnection);
            con.Open();

            OleDbCommand odc = new OleDbCommand(strQ1, con);
            OleDbDataReader read = odc.ExecuteReader();
```

```
            while (read.Read())
            {
                wordText = Convert.ToString(read["WT"]);

                OleDbCommand com = new OleDbCommand(strQ, con);
                OleDbDataReader dbr = com.ExecuteReader();

                while (dbr.Read())
                {
                    word = Convert.ToString(dbr["W"]);
                    if (wordText == word)
                    {
                        write.WriteLine(word);
                        //flag = true;
                        break;
                    }
                }
            }
            write.Close();
        }

        private void btnFind_Click(object sender, EventArgs e)
        {
            string strFind = "";
            string str = "";
            bool flag = false;
            int i = 0;
            string s1 = "";
            string s2 = "";
            bool f = false;
            string s3 = "";

            string s5 = "";
            string strQ = "";
            bool f1 = false;
            bool f2 = false;
            string str1 = "";
            string str2 = "";

            DateTime dateStart = DateTime.Now;

            strFind = Convert.ToString(textFind.Text);
            string[] wrds = strFind.ToLower().Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
            foreach (string s in wrds)
            {
                row = tQuery.NewRow();
                tQuery.Rows.Add(row);
                row["WordQuery"] = s;

                i++;
                if (i > 1)
                {
                    flag = true;
                }
            }
            if (flag == true)
            {
                DataRowCollection r = tQuery.Rows;
                for (int j = 0; j < tQuery.Rows.Count - 1; j++)
                {
                    DataRow ro = r[j];
                    DataRow ro1 = r[j + 1];
                    s1 = ro["WordQuery"].ToString();
                    s2 = ro1["WordQuery"].ToString();
                    StreamReader sr =
File.OpenText(@"D:\08.05.22\Slovari\МорфолСловарь\морфСлов.txt");
                    while ((s3 = sr.ReadLine()) != null)
                    {
                        string[] w1 = s3.ToLower().Split(new char[] { ' ', ',', '.', '(', ')',
';', '"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                        foreach (string s4 in w1)
                        {
                            s5 = s4;
                            break;
                        }
                        foreach (string s6 in w1)
                        {
                            if (s6 == s1)
```

```
                            {
                                str1 = s5;
                                f1 = true;
                            }
                            if (s6 == s2)
                            {
                                str2 = s5;
                                f2 = true;
                            }
                        }
                        if ((f1 == true) && (f2 == true))
                        {
                            strQ = str1 + " " + str2;
                            f = TextIndexMng.Instance.ReturnName(strQ);
                            break;
                        }
                    }
                }
            }

            if (f == false)
            {
                StreamReader sr =
File.OpenText(@"D:\18.10.22\Slovari\МорфолСловарь\морфСлов.txt");
                while (( s3 = sr.ReadLine()) != null)
                {
                    string[] w1 = s3.ToLower().Split(new char[] { ' ', ',', '.', '(', ')', ';',
'"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                    foreach (string s4 in w1)
                    {
                        s5 = s4;
                        break;
                    }
                    foreach (string s6 in w1)
                    {
                        if (s6 == strFind)
                        {
                            TextIndexMng.Instance.GetName(s5);
                        }
                    }
                }

            }

            StreamReader stream = File.OpenText(@"D:\18.10.22\Indeks\Query.txt");
            while ((str = stream.ReadLine()) != null)
            {
                row = tIndexText.NewRow();
                tIndexText.Rows.Add(row);
                row["TextName"] = str;
            }
            i = 0;

            gridBazis.Visible = true;
            gridBazis.DataSource = tIndexText;
            f = false;

            DateTime dateEnd = DateTime.Now;
            TimeSpan t = dateEnd.Subtract(dateStart);
            MessageBox.Show("Пошук виконано за "  + t.TotalSeconds +  " с");

        }

        private void підключитиСловникСлужбовихСлівToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Stream myStream;
            op.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";

            if (op.ShowDialog() == DialogResult.OK)
            {
                str = "";
                if ((myStream = op.OpenFile()) != null)
                {
                    using (TextReader strRd = new StreamReader(myStream))
                    {
                        string strW = "";
                        while ((strW = strRd.ReadLine()) != null)
                        {
```

```
                            SlugWord sw = SlugWordMng.Instance.NewItem();
                            sw.Name = strW;
                            SlugWordMng.Instance.Save(sw);
                        }
                    }

                    myStream.Close();
                }
            }
        }

        private void підключитиСловникиТематикToolStripMenuItem_Click(object sender, EventArgs e)
        {

            int countInDictionary = 0;
            int countAdded = 0;
            int countWordAdded = 0;
            int idWordSlug = 0;

            DateTime dateStart = DateTime.Now;
            Stream myStream;
            op.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";

            if (op.ShowDialog() == DialogResult.OK)
            {
                bool flag;
                str = "";
                if ((myStream = op.OpenFile()) != null)
                {
                    string fileName = op.SafeFileName;

                    FileInfo f = new FileInfo(op.FileName);
                    Dictionary dic = DictionaryMng.Instance.GetItemByName(f.Name);
                    if (dic == null)
                    {
                        Dictionary newd = DictionaryMng.Instance.NewItem();
                        newd.Name = f.Name;
                        DictionaryMng.Instance.Save(newd);
                        int idDic = DictionaryMng.Instance.GetIDByName(f.Name);

                        using (TextReader strRd = new StreamReader(myStream))
                        {

                            string strWS1 = "";
                            while ((strWS1 = strRd.ReadLine()) != null)
                            {
                                string[] wrds1 = strWS1.ToLower().Split(new char[] { ' ', ',',
'.', '(', ')', ';', '"', '-', ':' }, StringSplitOptions.RemoveEmptyEntries);
                                foreach (string s1 in wrds1)
                                {
                                    flag = DicContentMng.Instance.FindItem(idDic, s1);
                                    idWordSlug = SlugWordMng.Instance.GetIdByName(s1);

                                    WordDic w = WordDicMng.Instance.GetItemByName(s1);
                                    if (w == null)
                                    {
                                        WordDic newW = WordDicMng.Instance.NewWord();
                                        newW.WordName = s1;
                                        WordDicMng.Instance.Save(newW);
                                        countWordAdded++;

                                        if ((flag == false) && (idWordSlug == 0))
                                        {
                                            DicContent newCon = DicContentMng.Instance.NewItem();
                                            newCon.IDDictionary =
DictionaryMng.Instance.GetIDByName(f.Name);

                                            //newCon.IDWord =
WordMng.Instance.GetIDByNewName(s1);
                                            newCon.WordName = s1;
                                            newCon.IDWord =
WordDicMng.Instance.GetIDByExistName(s1);
                                            DicContentMng.Instance.Save(newCon);
                                            countInDictionary++;
                                            countAdded++;
                                        }
                                    }
                                    else
                                    {
```

```
                                        if ((flag == false) && (idWordSlug == 0))
                                        {
                                            DicContent newCon = DicContentMng.Instance.NewItem();
                                            newCon.IDDictionary =
DictionaryMng.Instance.GetIDByName(f.Name);

                                            //newCon.IDWord =
WordMng.Instance.GetIDByExistName(s1);
                                            newCon.WordName = s1;
                                            newCon.IDWord =
WordDicMng.Instance.GetIDByExistName(s1);
                                            DicContentMng.Instance.Save(newCon);
                                            countInDictionary++;
                                            countAdded++;
                                        }
                                    }
                                }
                            }
                        }
                        myStream.Close();
                    }
                }

                DateTime dateEnd = DateTime.Now;
                TimeSpan t = dateEnd.Subtract(dateStart);
                MessageBox.Show("Дані оброблено за " + t.TotalSeconds + " c" +
Environment.NewLine
                                + "Проаналізовано " + countInDictionary + " записів "
                                + Environment.NewLine + "Додано " + countWordAdded + "нових слів"
                                + Environment.NewLine + "Додано " + countAdded + " слів в
довідник");

            }
        }

        private void розробникToolStripMenuItem_Click(object sender, EventArgs e)
        {
            About a = new About();
            a.ShowDialog();
        }

        private void підключитиСловникДляМорфологічногоАналізуToolStripMenuItem_Click(object
sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            string textName = "";


            textName = TextMng.Instance.GetName();

            StreamReader stream = File.OpenText(@"D:\18.10.22\Indeks\I\"+ textName+"");
            while ((str = stream.ReadLine()) != null)
            {
                row = tIndexText.NewRow();
                tIndexText.Rows.Add(row);
                row["TextName"] = str;
            }
            i = 0;

            gridBazis.Visible = true;
            gridBazis.DataSource = tIndexText;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string textName = "";

            textName = TextMng.Instance.GetName();

            StreamReader stream = File.OpenText(@"D:\18.10.22\Indeks\II\" + textName + "");
            while ((str = stream.ReadLine()) != null)
            {
                row = tIndexText.NewRow();
                tIndexText.Rows.Add(row);
                row["TextName"] = str;
```

```
            }
            i = 0;
            gridBazis.Visible = true;
            gridBazis.DataSource = tIndexText;
        }

    }
}
```